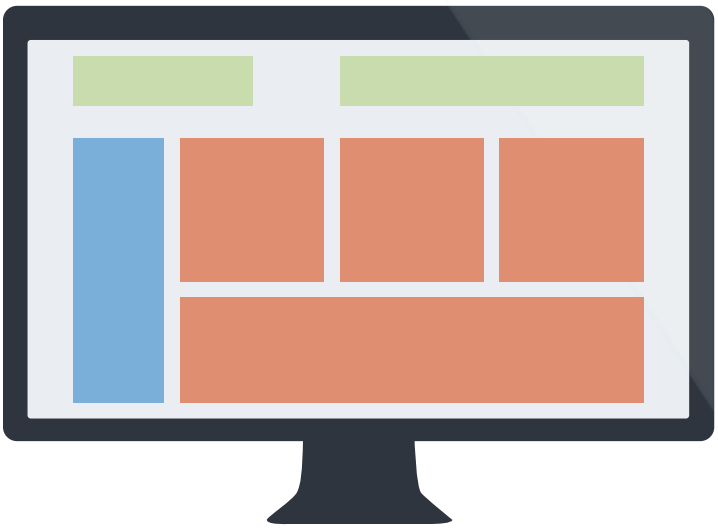
CHALMERS



Designing great developer sites

A study on how to approach creating intuitive and efficient developer resources.

Master's Thesis
Interaction Design & Technologies

ARON CEDERCRANTZ FIONA ROLANDER

Department of Applied Information Technology
Chalmers University of Technology
Gothenburg, Sweden 2013
Report number 2014:008
ISSN 1651-4769

Designing great developer sites
A study on how to approach creating intuitive and efficient developer resources.
Aron Cedercrantz & Fiona Rolander

Report number 2014:008
ISSN 1651-4769

Department of Applied Information Technology
Chalmers University of Technology
SE-412 96 Gothenburg, 2013
Sweden

Abstract

Developing software has become a wide spread occupation both as employment and for entertainment purposes. There are estimately 18.2 million software developers globally and this number only increases, partly thanks to improved and lowered pricing of technology. We see a trend in developing using others' existing libraries or platforms and thus also using the pertaining documentation guidelines to access and understand how to go about this approach.

The aim of this report was to investigate how developer documentation and guidelines on sites could be improved for the target group of developers. We looked into what characterized developers, current implementations of developer sites and thereby presented a set of recommendations on things to have in mind when designing their interfaces. The report will cover the theoretical approach, the process behind the results and a case study prototype through design driven research.

Conclusions in this report will help designers when approaching designing a developer platform site in regards to understanding the users and key points to have in mind. The report will discuss broadly on the dos and don'ts and argue both ways in order to fit a general approach and not merely our case.

Keywords: web design, interface, interaction, API, developer, Spotify

Acknowledgements

The authors would like to thank the following for contributing to this project; Thommy Eriksson at Chalmers University of Technology, Henrik Eneroth at Antrop AB and the Community squad at Spotify AB. We were warmly received at Spotify and the help and support has been great. A thanks also goes out to all the people who participated in the project via interviews, observations and those who contributed their knowledge in some other way.

Contents

1

Introduction

We find ourselves in a time where technology has been made easily accessible to every ordinary person. With the increased set of tools follows the increased amount of opportunity to create things on one's own. The number of developers in the world today has become a large, and fast growing, amount. For example in the US there are about 275,000 registered developers just for iOS (Apple Inc, 2013) and an estimated 18.2 million software developers globally which is expected to grow to 26.4 million by 2019 (Patrick Thibodeau, 2013). This trend implies a growth in development occupation and interest which subsequently means approaching it both professionally and for the sake of enjoyment. A developer site is generally a website with a set of tools, guidelines or resources to assist in creating new software. It often manifests as a company that provides the possibility to incorporate their functionality into other software through an application programming interface.

The increase of developers and their ways of developing requires a wide set of tools and guidance. We expect developers to create great things, but who creates the tools to help them create great things? At the time of writing we have found little research on the subject of how to design intuitive documentation for developers.

We researched the field of web design and developers as a target group. The final concept shows how the user experience can be improved when using developer sites before or in the process of creating a piece of software.

This thesis was performed at Spotify AB in Stockholm in 2013. The developer site at Spotify was used as a reference project in order to achieve a design-driven approach where a prototype of an improved design for this page served as support in drawing general conclusions.

1.1 Intention

Developer resources generally contain a vast amount of information, most often presented in text. This stands out from general web experiences that tend to be more concise and

product or service oriented. Due to the increase in amount of developers, resources and information we see a trend in escalated complexity when at the same time there is little progress within usability research in this area.

The intention of this thesis is to study how design theory can contribute to improving the experience for developers when using developer resources. This incorporates investigating and expanding on user psychology of developers, web design theory targeted for them and the context in which it takes place. Furthermore the goal is to research ways of improving existing documentation based on organization theory and structure design. By studying interaction flows we aim to make it easier for a developer to absorb and understand the content of a site.

In order to fulfill the intention we need to define what a good experience for developer implicates and then how to achieve it. We ask ourselves the following question:

> *How does one design a good user experience for developers in regards to developer websites?*

Answering this question will give us an understanding of the target users and help produce a set of requirements that will tailor to their needs. It also means creating a process towards achieving that. We hope that the results derived from answering this question will provide the reader with guidance of how to approach a similar case by either adopting our process or customizing the results to fit her purpose.

We focus on presenting a set of guidelines applicable when designing developer sites through a design driven approach on Spotify AB's developer site. The aim is to define factors which contribute to a developer getting a good understanding of that system and help provide a good experience.

1.2 Delimitations

Even if a design would be perfect it would not compensate for bad content. Site content must be correct and useful. We will however not discuss content in our project since it is outside our scope. If one wants to know more about this then we recommend starting off by reading *Measuring API usability* (Clarke, 2004).

As stated in the research question we intend to look into and answer how one designs a good user experience in the context of a developer website. Since the environment usually is a known company and distributed on the Internet we estimate that apart from developers also might target a wide range of different visitors such as consumers, entrepreneurs, managers and so forth. We will only focus on the needs and behavior of developers and research how to improve the user experience for them. Should a site be developed for commercial or business purposes then we recommend developing personas for *all* relevant users and revise priorities for the e.g. the landing page.

In this project competitive analysis was carried out and the results from that are relevant to how the competitors developer sites were formed at that point in time. We do not take responsibility for observations which might not coincide with the state of the sites at the time of reading this. For cross comparison we refer to the Internet Archive (https://archive.org/).

An important part of a developer site might be managing one's own content through a console when being logged in such as managing released apps, files or bookmarks. We will not present design of this since it is dependent of each site's purpose and requires a high level of detailing which is not as relevant to this study.

2

Theory

This chapter presents the underlying theories to this project. It covers topics relevant when designing for the web such as layout design, user studies, information visualization, aligning design throughout products and thinking forward in interaction implementations. Also relevant to the project is benchmarking competitors' sites and analyzing the status of the current design of Spotify's developer site which is also presented in this chapter.

2.1 Current condition analysis

In order to look forward we start by looking at the present. Conducting a thorough site review is a good start toward identifying the possible problem of that existing design (Bowles and Box, 2011). This section describes the theory behind analyzing the quality of the current page and what can be learnt before a future design approach.

2.1.1 Information architecture

Information architecture can be defined as "the combination of organization, labeling, search, and navigation systems within web sites and intranets" (Morville and Rosenfeld, 2006). The field of information architecture and interaction design is believed to coincide more and more. Information architectures is even by some concidered a subset of interaction design (Goodwin, 2009). Designing a good taxonomy is the base of almost all information architectures (Morville and Rosenfeld, 2006). With large sites containing a lot of content on subpages visualizing the relationships between pages can be done by a hierarchical model where it shows parent-child linking. In order to more easily study, and remodel, the balance between width and breadth on a site this model is very useful. This top-down approach helps designers or information architects count number of clicks from point A to B, grouping of item categories and thus requirements in the design. Since the aim of this thesis discusses intutitive organization of information on the web and designing an understandable flow it seems natural to start off by studying the

current information architecture of the site. In order to more easily grasp the entirety of a system, such as a developer site, it can help to plot out the content's structure and the access behavior patterns (Cooper, Reimann and Cronin, 2007).

2.1.2 Site analytics

Website analytics can be used as a an additional tool, besides for example interviews and direct observations, for collecting data on a given set of users. One can see it as a form of hidden observation which is always looking at what the users are doing, in what sequences, how they got there and a lot more[1]. Different analytics applications and services often provide a lot of similar metrics such as number of visits to the site or separate landing pages, bounce rate, what is being clicked, the flow of visitors and so on. One of the primary drawbacks of this form of data collection is that it is dependent on which application or service is selected for the task as they can provide very different types and amount of data. The level of analysis the different provide might also vary.

Furthermore these tools depend on the existence of a website before they can show any results. These kinds of tools are thus only applicable if an existing site is available or one has access to a third-party site with the same target group. Given that one has access and can collect analytics from such a website it provides an easy as well as convenient way to collect data (Hasan, Morris and Probets, 2009) which can then be used to enhance the understanding of one's target group. Although it depends on which analytics service is used and what it offers in terms of automatic analysis it can become quite time consuming to analyse the collected data which Hasan et al. also writes about.

The data collected via an analytics tool can then be used to inform the design process by highlighting the current visitors' behavior. One can for example figure out how, from where and why a person ended up on the site. Be it via another site (hereforth also referred to as a "referral"), a search engine, direct traffic where the user has entered the address of the site or some other in-vector such as e-mail. By analyzing the in-flow of traffic and combining it with the landing and exit pages as well as, when applicable, the search terms used it is possible to get a somewhat decent picture of the visitors' interaction with the site. What content and resources they are interested in as well as which pages they actually visit, then the correlation of sought content and visited pages should be kept high.

Another metric which is related to how and where visitors end up on the site is what they do next. After entering the site there are primarily two alternate scenarios which can occur. The visitor could either continue browsing more pages on the site or leave without viewing any other page. Each of these tell their own story and can be hard to interpret on their own as each action might be a negative or positive event. It depends on more factors such as whether the user found the sought content or not. As a consequence other

[1]There are too many different types of metrics which can be observed to mention in this report and is thus outside our scope.

types of studies might be required to fully understand the flow whether it is viewing another page or leaving the site (i.e. a bounce, which will be described below in the section on bounce rate).

2.1.2.1 Bounce rate

An important aspect in relation to the visitors' flow and behavior for which developer sites might differ from the more common target of literature is in regards to bounce rate. A high bounce rate is often seen as a negative factor (Sculley, Malkin, Basu and Bayardo, 2009; Bowles and Box, 2011). A rather large portion of literature and help texts on the Internet in regards to website analytics focus on e-commerce and marketing sites where some form of conversion is sought. Such a conversion could be a purchase of some goods or service, or that the visitor registers for an account. For these types of websites it is often desirable to have a single user visit several pages on the site during a single session as it could increase the likelihood of the conversion happening due, or thanks, to prolonged exposure. A too large amount of navigating around could also be an indication of a problem (Bowles and Box, 2011). On the other hand, a developer site can in certain cases serve a different role where a higher bounce rate and shorter average visit duration is seen as something positive.

The reason for why a higher bounce rate can be interpreted as a positive aspect instead of a negative one in regards to a developer focused website is that a large portion of its pages serve to supply the visitor with information, a specific answer, some resource or similar (henceforth these will be referred to as a *resource*). This means that if the user enters the site on the same page as the sought resource is presented on and only spends a short amount of time there followed by directly exiting from the same page then it should be interpreted as a positive event.

There are some exceptions to these guidelines on bounce rate. First of all, if the visit duration is too short such that it becomes clear the visitor has not been able to attain any information or resource, a bounce means she decided that the page did not contain the sought resource. What this threshold is depends on the content on the specific pages. Furthermore if the bounce rate creeps up to high it could also be a sign that something is not right as you would expect a few visitors to still continue browsing the site for more information. To provide a more certain answer whether it should be interpreted as a positive or negative event one needs to study the user's intent behind the behavior. This is discussed in section 2.2.

The clear exception in regards to the bounce rate however is pages such as the home page (given that the entire site does not consist of a single page) for which one of the most important aspects is to serve as funnel routing the user to what she is seeking (Nielsen and Loranger, 2006). This funneling should be completed as quickly as possible and with an as low bounce rate as possible. In the research by Nielsen and Loranger they found that an experienced user only spends an average of 25 seconds on the home page

while a person less experienced with using and browsing the Internet might stay for a bit longer with an average of 35 seconds. A high bounce rate on such a page, void of any real information, can only be seen as a negative event. As Kaushik (2007) from Google wrote about the users who bounce;

> "They came, they said yuk and they were on their way."

This would mean that the home page does not fulfill its design goals which will be discussed in more detail in the home page theory section.

As one of the primary purposes of a home page is to funnel users to the resource they are looking for, it is advisable to look at what and where they click. By using such data, problematic page layout issues can be identified. As an example, if a link is often clicked but for the average visitor it would only show up below the fold[2] it might be reasonable that it should be moved. This data can be attained by using eye tracking or logging of where the user has clicked and correlate it with page elements.

Such direct data can be hard to get due to eye tracking being quite expensive (Ross, 2009; Pernice and Nielsen, 2009), requiring the user and researcher to be at the same location, being time consuming (Pernice and Nielsen, 2009) and finding services that provide the correlation between clicks and elements. An alternative is to look at the visitor flow to at least get an idea of which links are used by the visitors. The same could also apply for the opposite where a seldom clicked link or call to action is placed very prominently on the page. Using such information the page can be laid out with the content the user finds the most interesting in an as easy to get to way as possible. All of this also applies to deeper pages although the focus might be oriented differently on them. One should therefore not blindly apply the insights gained from this analytics data without considering the context of the page.

2.1.2.2 Location, date and time

The location from where the users of the site visit as well as which time of what day can provide information about what types of persons, and in which contexts they are visiting the site. Especially if one mixes in the properties of the device used when visiting the page (which is discussed further in section 2.1.2.3, " device properties"). Given a developer site it is reasonable to consider at least two contexts due to the duality of many such sites; used by a persons visiting the site while working or by a person in her free time. Depending on the distribution of visitors over days in the week and at what time, a rough estimate can be used to decide on persona prioritization. This depends on what the developer site's main target is as well as the reach of the site. One should take care to not draw any conclusion concerning the experience level of the visitors based on this data as it only provides a hint at the context of use by looking at the trends.

[2]Section of a page which is outside of the browser viewport and thus requires the user to scroll for the content to be visible.

Similarly the location and language of the user and whether it affects their behavior when visiting the site can also be of interest. For example, if the bounce rate, visit duration or other metrics is higher or lower for visitors of a certain language or location it could need further analysis. Such a trend could be a sign of interest in the offerings but not understanding it due to language or cultural barriers. Alternatively the site could be performing badly in the region or they could for some reason be ending up on the site without the intention. Again an absolute conclusion can be hard to draw from these metrics, nevertheless they can provide a good indication of matters which might need further investigation.

2.1.2.3 Device properties

Site analytics can also inform the design process in terms of constraints regarding what technology can be used and the visitors device properties. Many website analytics services collect not only which pages, session duration and so forth but also what the visitors use to view the website. Some of the metrics which can be, more or less, directly applied in the design phase concerns the properties of the users' screens. More precisely its resolution, aspect ratio and color depth. The distribution of resolutions could be used to decide which resolution classes (smartphone, tablet, laptop or desktop sized screen resolution) should be focused on. It is generally estimated that only 23 % of the home page visitors scroll the page (Nielsen and Loranger, 2006). This percentage then decreases even further for each subsequent visit. It might therefore be advisable to layout important funnels above the fold on the home page especially when also considering the short amount of time spent on said page. This is not to say that all content should be available above the fold but rather that the most important elements should be there, which these are and how to prioritize is discussed in section 2.7 on the home page.

An additional aspect concerning the screen resolutions is in regards to the distribution. Whether they are scattered all over the spectrum with somewhat equal amounts of visits via devices with a low, medium and high logical[3] resolution or aggregated around a single resolution group will affect the design approach. The former requires a much larger focus on making the site responsive in concerne to resolution while in the latter case it might not be an as important factor.

The data also contains information on the visitor's operating system and browser which can, and most likely should, be used to decide upon how a feature is implemented. As well as whether certain features can, or should, be utilized. For example depending on the combination of devices, browsers and operating systems it might not be a good idea to use a custom font. A more basic typeface might for instance be more appropriate as the average user's operating system and browser combination is unable to render more

[3] A logical resolution differs from the actual resolution of the screen in that a logical resolution of 1024 by 768 *points* can be represented by several different "real" resolutions. For example a logical resolution of 1024 by 768 *points* at double the dots-per-inch, DPI, would be represented by 2048 by 1536 actual *pixels*, or four pixels to a point.

advanced typefaces in an acceptable fashion. What these constraints are in actuality depends greatly on the traffic which is seen.

2.2 User study

According to author Jenifer Tidwell, the first step toward creating an interface is to understand the purpose and goal of its user (Tidwell, 2011). But in order to understand their goals, we must of course first know who the users actually are. Thus, in the process of creating a product the first natural step is understanding people and identifying the relevant audience for our goal. Understanding people means understanding the what, why and how about their personality. By knowing your user you are able to design a better product for them, thus creating a better experience alltogether. Since the user interface conveys the communication between human and machine it is important that it helps the user achieve the goals they set out to accomplish.

> "If any of the products that we use fail to work in the way that we expect them to, we generally label this as poor design or usability." Clarke (2004)

In this chapter we try to establish what the users are like, why they want to use our products, and how will they interact with it. Therefore, we set out to identify our audience. In order to identify our audience we get in tune with basic user psychology studies.

2.2.1 User Psychology

Two of the main approaches to theoretical psychology studies are cognitive and social design (Leech, 2013). When designing user interfaces we mainly focus on the cognitive which targets the processing and storing of information. Questions like "how fast will the user find this?", "how easy is this understood by the user?" belong within this topic. However, Joe Leech warns us about analyzing this behavior without taking into consideration the surrounding factors which might affect the outcome. He points out that when studying cognitive psychology in humans we often forget just that, that we are humans. Humans that deal with mutliple parallell processes at a time instead of focusing our total attention to a single entity. Although this might not have a huge impact on our design work or the way we come to conclusions, it can be good to keep in mind.

One can approach a proposed solution to effective developer page design with a cognitive psychology study in mind. It helps one to predict how future users will behave. If enough behavior of a typical user can be mapped, then one can presume that the next one will behave similarly. Further one can also apply this to assume how users will interact with a specific design in mind. This is called prescriptive psychology. Leech compares this to descriptive psychology theories which instead are based on a large amount of studies and can show relationships between arguments. However, descriptive psychology

lacks answering the question "why?". Descriptive psychology can be good for creating a high-level goal or path for designing. But in order to solve problems that can arise when designing for humans, prescriptive psychology theorizing is key. We could do fine with learning what the users do, but without getting to the root of why they do it we will never be able to design for future behavior.

2.2.2 Defining the User

It is easy to assume that all users have a mindset equal to our own and that our opinions will be reflected in them. This is wrong. *"Know thy users, for they are not you."* (Dupuis, 2003). Furthermore, one user is not equal to another user. But if we were to design for every aspect of every user, we would never come to a final design[4]. The aim is to draw a conclusion about what generally applies to the general users, which means identifying similar patterns among several users and ruling out the small differences which separate them. By mapping behavior patterns we can try to position ourselves in their way of thinking and thus easier establish if our product will meet their needs. This immersive procedure is known as *qualia* (Mills, 2011). Qualia means trying to experience something as if you were in someone else's shoes. It helps understanding our user significantly. Defining and designing for our subset of users is key since, as Kardy points out, "there is no such thing as a neutral design" (Dennis Kardys, n.d.) and we can't please everyone.

So then, who are our users? There are multiple ways of finding out who your target users are. Some research topics on collecting data about them include case studies, surveys and direct observations. How data is collected evidently affects the outcome of the result. It is important with well-nuanced, sufficient amount and relevant data. After this collection, creating a so called persona is a good way to represent data. Creating personas is a well-used technique when it comes to carrying out user-centered design (Cooper et al., 2007). It consists of creating a fictional character of a typical user based on the data that has been collected. A persona is a great tool for making design decisions based on the users' needs and not one's own. Since personas are created based on real data we ensure for credibility and correct portrayal. Personas also help create a more versatile approach to producing the design since it gives the designers more of a storytelling behind each user goal (Mills, 2011). Instead of User A wanting to get to point B, instead we get Simone trying to find information about directions to a mexican food restaurant. It opens up more of a discussion and possibility to create scenarios and formulate questions easier. The user could then be Simone, a compilation of the many real users met along the way. Furthermore, a theory is that learning about softer values such as beliefs, values, etc. might open up for more discoveries that the user has not thought of to express herself.

[4]Albeit, there is actually no such thing as a final design.

2.2.3 Data Collection

Data collection within the academic world has had its main focus on quatitative research (Cooper et al., 2007) which means collecting large amounts of wide data and presenting the overall result. This is based on the theory that "numbers don't lie". But in fact, they can. Statistics can be interpreted differently depending on their presentation and thus possible to modify it so that the viewer sees what you want them to see. Therefore they can not always be relied on. Presenting values and numbers in charts will not make it easier for one to understand the human behind the research data. An analysis of a human will probably contain a wide range of parameters and therefore raise the risk of chart junk (Suda, 2010) which makes it more difficult for the viewer to understand the purpose and information in a graph. So in a way we must separate complex human activity research from highly scientific research. Qualitative research can help us to understand that extra more about the user; contexts, behaviors, attitudes, social aspects, product usage, goals, dreams, burdens, etc. (Cooper et al., 2007) Quantitative data must be supported with qualitative research.

2.2.4 User research

As our research questions states we aim for a study for developers. Thus a study on what characterises a so called "developer" is in place. It is not a new realization that in order to improve the outcome of the technology we must start by improving the tools and studying their cognitive effects (Sajaniemi, 2008). PoP [5] studies show to be very extensive and varying in terms of level and task. One should therefore aim to target one's research to a relevant subset.

The Visual Studio Usability group defines three groups of developers; the opportunistic, the pragmatic and the systematic (Clarke, 2004). They then reach the conclusion that design should be approached in a scenario-based manner. Furthermore they point out the importance of a common vocabulary when discussing developer tools. In other words, content must be adapted to fit the developer mindset. Since we do not dig deep into the actual formulation of text we apply this to a high-level mindset of the developer, meaning navigation and browsing manner.

The description of the developer stereotype in literature portrays a progress-kean person. They praise efficiency (Bowles and Box, 2011) since they themselves try to minimize the effort they put in. They balance high quality with effectiveness and time resources. Decisions are not made lightly, they need a solid background and explanation. However, reading about stereotypes are as versatile as reading about chefs. There are all kinds, from heating a microwave dinner to a michelin-star chef. This is why we turn to observational studies to further deepen our understanding of the users relevant to our page.

[5] Psychology of Programming

2.3 Information visualization

When designing a site, or anything for that matter, which will visualize some sort of information one must take into account at least a few key elements. Whether or not elements can be interacted with needs to be clear (Krug, 2006). The field of information visualization covers topics like semiotics, color, patterns, visual attention, item clustering, imagery and so on (Ware, 2004). This section will discuss two areas which are especially relevant to website design. Namely colors and typography as these make up a large portion of most sites. Also, the areas of layout and navigation is discussed in site elements (2.6) while clustering is mentioned in the section on information navigation (2.1.1).

2.3.1 Colors

Our eyesight is the most important of our senses when it comes to branding and marketing. Subconsciously we judge a product in 90 seconds and more of half the evaluation is based on the color (Mills, 2011). Brand recognition is increased up to 80 % with colors (Morton, 2010). It becomes even more significant that visuals are important when designing for the web as there is little usage of the other senses. Colors evoke feelings and behaviors because of memories attached to them. Furthermore color can be used for attention attraction, element grouping, navigation aid, storytelling, cultural dependant communication, memory improvement, etc. Color choice can therefore have a direct impact on the user experience.

> "Choosing colours is not a decision to be made lightly." Suda (2010)

When there is a lot of content that cannot be processed at the same time color is a way to highlight a certain part more than another (Suda, 2010). This fictive hierarchy is done by directing the users attention to a grouping of content.

The idea here is not to dive deep into the world of color science but instead to have it in mind in order not to compromise the user experience. Choosing your colors wisely and letting them be a part of the design process. It will pay off.

Choosing colors for a dev site, or any other site, can depend on the following factors:

Context
Different colors can be used in different contexts. For example red is a color which can be useful for important calls to action, such as "click here" or "buy now", or warnings because of its high visibility. Depending on the tone, blue can come across as cold and unwelcome or loyal and political. Green is associated with concentration, environmental and wealth. And like this it goes on. But how does it affect designing a developer page? Depending on what we want to communicate, color selection can be difficult but effective. Reflecting on how your color selection affects the overall feel of the page and attention focus of the user is a valuable step in the process of designing.

Culture
Color meaning varies a lot depending on what country we are referring to. Between the East and the West we find vast differences in color culture (Mills, 2011). So how does this affect our choice of colors? Know your audience. From where do they come? Which do we want to focus on? It is important to understand what the culture behind your target user is to define relevant differences.

Brand
Just as in our case, many developer sites are linked to a company or brand. This often means choosing your colors depending on the brand guidelines produced by that company. Brands tell stories. And when it comes to designing for a brand, consistency is important. The relationship between brands and colors can relate to deep inward feelings rather than to the intellect. (Suda, 2010) Choosing colors according to brand is key in order to convey the company character and communication purpose.

Technology
As previously mentioened, different screens come in different sizes, resolutions and prices. In order to get a good color transition, millions of colors are needed (Ware, 2004). Although the average screen of today has good support for different color depths and contrasts it is good to take into account how monitor gamma values can affect the look of your design. Furthermore, should there be a need for printing the dev page, have in mind how costly it may become with a heavy amount of ink. Provide the user with a printer-friendly version (Travis, 2009).

Color blindness
Impaired color vision is found in as much as 10 % of the male population and 1 % of the female. (Ware, 2004) There are three common versions of color blindness to be aware of; deuteranopia, protanopia and tritanopia (Suda, 2010). Thus, choosing colors suitable for them is preferable. Since color blindness often takes its form in the red-green area you must look how they are perceived when they see it. Redesigning in blue-yellow is an option or making sure the shade of green-red will be distinguishable enough for color blind people.

2.3.2 Typography

As with color it is important to consider the typography of the developer site as it will not only affect the coherency and its connectedness with the organization behind it but also the legibility and usability of the site. As discussed in the section on branding (2.4.1), by aligning the typography to the rest of the organization a sense of trust can be evoked where the visitor does not need to worry or think about who is delivering the content. Less time is then spent on figuring out the unessential information outside what she was searching for. Before delving deeper please be aware of the difference between a typeface and a font, the former is the concept while the latter is a specific realization of a typeface.

In terms of legibility and usability the choice of font matters as different fonts are suitable for different types of elements (such as body text, headings and so forth) (Brown, 2013). As a consequence it is important to think how the brand and feeling of the organization can be combined with the needs and requirements of the developer site. Furthermore the tone and style of the typeface should be considered such that it match how the site wants to pose itself. That is, should it feel open, informal and welcoming or should it perhaps be more strict, authoritative and formal? As an example many *sans-serifs* feel informal and approachable (Mills, 2011). There is no readability difference between the two major classes of typefaces, namely*serifs* and *sans-serifs*, but it can differ between specific fonts (Bernard, Liao and Mills, 2001). Additionally the typeface used in special cases such as for code listings needs to be picked such that it works for showing off code while also matching the base typeface.

Another important aspect for readability is the point size used as well as line height[6] and length, especially when it comes to the body text. Most studies seem to suggest that a font size larger than 12 points is easier to read and understand (Bernard et al., 2001; Nielsen, 2002). Although these studies do target an older population which might not coincide with developers it would seem reasonable that it could apply to younger people as well. Reichenstein (2006) proposes that at least 16 points is used as the minimum body text point size on a screen as it matches the size of 12 points printed on a paper held at an appropriate reading distance in front of the screen. The reason why the point size should be larger on a desktop screen stems from that the screen is further away from the eye of person reading than the paper sheet.

Only adjusting the point size might not be enough as the height and length of the text lines also matters for readability. For example, when the length of the lines increases the text becomes more difficult to read and comprehend. To counteract this problem the line height should also be increased, else the eyes will find it hard to return to the start of the next line (Dyson, 2004). One method to determine a good mix of these three properties is using the golden ratio as proposed by Pearson (2011). This leads to four different formulas, where the first two define the line height (2.1) and length (2.2) based on a point size while the third gives the line height based on point size and line length (2.3). The fourth and last equation provides the line length given a point size and line height (see 2.4). Due to the nature of screens technology the result of the equations should be rounded to the nearest integer. In the equations below (2.1, 2.2, 2.3 and 2.4) p references the point size and φ the golden ratio ($\frac{1+\sqrt{5}}{2}$).

$$h_{\varphi} = p\varphi \tag{2.1}$$

$$w_{\varphi} = (p\varphi)^2 \tag{2.2}$$

[6]The distance between two consequtive rows of text.

$$h = \varphi - \frac{1}{2\varphi}\left(1 - \frac{w}{w_\varphi}\right) \tag{2.3}$$

$$w = h_\varphi^2\left[1 + 2\varphi(h - \varphi)\right] \tag{2.4}$$

Pearson (2011) also adds a note "Factors such as x-height and other typeface metrics also influence typography and should be considered in finalized designs." or in other words, this method provides a good starting point which should then be tweaked optically (that is, so that it looks good).

2.4 Product based design

Often developer pages are not the core business purpose of organizations but rather a supplementary service. We therefore try to identify different relationships between developer pages and their related pages. In some cases the pertaining page is the core service, such as the relationship between facebook.com and developer.facebook.com. In other cases we identify stand-alone pages, such as developers.app.net, where the developer page actually is the foremost service. Lastly we identify developer pages which have related pages that market another separate product, such as Spotify.

Our case study focuses on the last mentioned, which we call a product based developer site.

2.4.1 Branding

A design of a website communicates invisible communication (Mills, 2011) and there are a couple of techniques that can be used to achieve this and subsequently enchance the user experience. Matching the look and feel of the brand with the website can be done by using appropriate colors for page elements like buttons, links, icons, headings, footers, etc. A text can convey a message, but so can the typeface it is written in. We make sure that to these contributes to a stronger brand expression. Depending on what mood the company, and thus brand, aims for should directly affect the tone of voice in the content. A playful page for kids might use informal wordings and rely more on imagery than text while a government page might do the opposite. The overall aim is to acheive a mood that is aligned with the company's style and invoke feelings and emotions. A sense of safety and reassurance that the user has arrived to the right page appeals to the empathisers while facts appeal to the systemisers (Leech, 2013). Feeling trust for the page can therefore increase the trust for the brand if they are well-aligned.

2.4.2 Alignment

If a developer page has a pertaining product page, such as most product developer pages have, we identify the need for comparing these two in order to check for alignment. We believe it to strengthen the feel of trust due to aligned branding. Further alignment between product and product page invokes concistency and reduces the overlap. However it must make sense to align these and not sacrifice usability for the sake of alignment (Cooper et al., 2007). When it comes to developer pages we also want to make sure there is no confusion between the product page. The user must understand that she is on the developer site and not the product site.

2.5 Creative design

From an inspirational talk on TED about rethinking how we approach design (Brown, 2009) we learn that we have previously thought about design as "converging" in order to *make choices*. Brown claims and advocates for exploring the field of "diverging" design where we instead aim to *create choices*. Applying this notion of combining both converging and diverging design to achieve creative thinking moving forward can open up for a great variation of new solutions. Since we limit our research to very technology savvy users it is reasonable to assume that they might be a good target group for adapting to creative interaction implementations. Developers develop. They are creators and thus take an interest in solutions which improve their ability to build things efficiently (Bowles and Box, 2011). Goodwin encourages designers to design for the long term, pushing the constraints to look ahead in time and create for the future (Goodwin, 2009). She also points out the need to balance the existing constraints with optimistic new approaches that encourage creativity. What we therefore learn is that we should not be afraid to take on new ways as long as it does not impair the original demands. If time allows for it, discussing and researching how to take the design one step further to cater for future needs might pay off. She also points out that it is sustainable to anticipate future changes that might take place and from the start adjusts to these. If the changes then take place the implementation would not need to be entirely reconstructed. Either implementing solutions that already prepare for future add-ons or that can be removed without affecting the whole of the experience might therefore be advantageous.

Although it is ideal to design for intermediate users (Cooper et al., 2007) we learn that with access to tools they are receptive to adapting to new features. Perpetuate intermediate users might not need or want to use advanced features but merely knowing that the extended feature exists is reassuring to them. Therefore an option can be designing creative solutions that are intended for experts but that intermediate users can *choose* to utilize. Balancing design which beginners can easily grasp, intermediates feel comfortable with and experts get challenged and inspired by is a challenge but a key to success.

2.6 Site elements

To facilitate our analysis we break the site down into a few site wide elements. In this section they will each be discussed in an in depth manner, detailing considerations one should have in mind when designing such an element.

2.6.1 Navigation

One of the most important elements of the site is the navigation which can not only take the visitor to where she wants to go but also provide an overview of the content available. More so the area of navigation also encompasses orientating the user and informing her how to use the site (Krug, 2006). All of this needs to be executed, in conjunction with the site's information architecture, such that it maps as closely as possible to the user's mental model and expectations of the site (Nielsen and Loranger, 2006). Great care must therefore be taken when designing and choosing between different forms of navigation. It does not only consist of a main menu but also the browser's navigational features such as the back and forwards actions and also other elements on the site such as breadcrumbs.

One of the primary data points which should be considered when creating the navigation is the amount of pages, their depth and also pages per depth as this will affect how it should be presented. If the amount grows unwieldy it can be a sign of problem with regards to the site's information architecture and that it should be dealt with before continuing with the navigation. Another important factor of the navigation is to keep it consistent and coherent throughout the site and all its pages as this helps the user orientate herself and understand what she can do as well as how to get to the next place (Nielsen and Loranger, 2006; Krug, 2006; Norman, 2008). If the navigation where to differ (be in a different location, be ordered differently or only show a subset of the options) from page to page this could cause confusion and as Nielsen and Loranger (2006) put it "people must shift their attention from using the site to figuring out how to use it".

To accomplish these goals it is important for the navigation to follow the hierarchy perceived by the user (Krug, 2006). An example would be the user trying to find information about a specific module of a product library's API (application programming interface). She could then, for instance, start by looking for the product and then the API reference or documentation and then the actual module. The order and hierarchy in which the user look for signs needs to be determined by some form of user studies which are discussed in section 2.2.

Once the size of the navigation has been established it is possible to start looking at how to fulfil the task of providing the user an overview of where she is, what is available, and how to get to where she wants.

First of all each of the site's pages must have a page title stating clearly where the

visitor is right now, preferably in large type and at the very top of the page (Krug, 2006). Secondly there needs to be something communicating where the page resides in the site structure. This could, for example, be achieved by utilizing a breadcrumb or through the menu. A breadcrumb element is a chain that represents the page's category or section and its parent categories. Each item in the breadcrumb should be actionable and take the user to that category. Which one to choose depends heavily on how the menu system will be designed as a requirement for the menu to work as context showing element is that at least all parental categories of the page is visible without any user interaction. If the menu system is used it is of extra importance that the cues to communicate which page is selected and which categories are its parents. This is also important when a standalone breadcrumb element is used, using multiple visual cues to indicate an active or selected state greatly helps to make it clear where the user is (Krug, 2006).

A menu system is in turn a very good way of communicating what is available and how to get to the next place. The design of such a system to a very large extent depends on the breadth and depth of the site, where one should aim for a broad and shallow architecture as possible (Norman, 2008). One of the most important aspects of the menu is that it the items are ordered in a natural order (such as alphabetically) as this leads to a logarithmic search time for the user, otherwise it becomes linear according to a study by Norman (2008). It can also be tempting to hide a complex site structure in one or more drop-down menus but this might be problematic as it would, first of all, hide the current state of where the user is (it could be alleviated to an extent using a breadcrumb, even though that should be considered a complement rather than a replacement). Secondly, using drop-downs leads to an increased complexity which is more difficult to control and makes it easier to get lost (Nielsen and Loranger, 2006). Instead it might be worth trying to display the entire structure at the same time, or at least the currently active path or context.

This in turns leads to the next decision of where to place the menu, or several menus, as well as how and when they should be visible (see figure 2.1, 2.2, 2.3 and 2.4). There are a few distinct locations possible; at the top, bottom, to the left or right of the content. Each of these come with their own set of advantages and drawbacks. The placements can then also be combined with a "floating" menu, that is stays at a fixed location whether the page is scrolled or not. Making the menu element float or not results in roughly the same benefits and disadvantages.

A floating menu has the advantage of always being in the same spot no matter how much or little the user scrolls a given page. This leads to the user not having to figure out what to do to get to the menu more than once. Combining a floating menu with a static menu which also shows the user where she is, like a breadcrumb, can lead to a combination which can successfully deliver on the requirements of the navigation system.

The primary drawback of such an arrangement on the other hand is that it can respond somewhat poorly if the size of the browser window is not large enough to

display the entire menu at the same time. It could in such a case lead to multiple scroll bars if the menu is on the left or right hand side of the content. If the menu is at the top or bottom it could also start obscuring too much of the content area to be feasible.

A stationary menu which scrolls away with the site's content as the user scrolls has the advantage of getting out of the way as the user deepens into the content. It also means that the menu can become more resilient and respond more naturally to the situation where the size of the user's browser window is smaller than needed to show the entire menu at the same time.

On the other hand, as the menu scrolls with the content the user might have to scroll back to the top or down to the bottom (depending on the menu's placement) to view it. Consequently losing her position and perhaps also goal.

In turn putting the menu at the top (see figure 2.1) could, depending on amount of pages, lead to either having to use some form of drop-down menu system or a system of at least two menus. The former alternative of a drop-down menu could lead to, as discussed in a paragraph above, visibility issues for the different options as well as control issues (Nielsen and Loranger, 2006). It could on the other hand be quite compact and work quite well as a floating menu. If there are only a few subpages it might work even better as all of those could be visible at all times. In the latter case, the suggested approach is one global menu at the top for each section using tabs and a second "local" navigation with the sub-sections and pages of the selected global section as proposed by Krug (2006) in his book "Don't make me think". Such a configuration has the benefit of making it clear in what section the user is. As well as reducing some of the space constraints for the local navigation. A possible downside to this arrangement is that the local menu could be completely replaced or even removed when visiting a different section, something which could be jarring or a cause for confusion (Nielsen and Loranger, 2006).

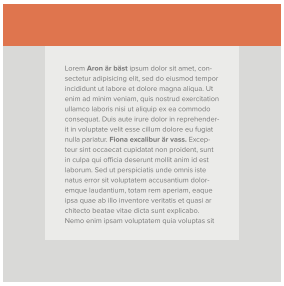


Figure 2.1: Menu at the top of the site.

Having a menu at the bottom of the site (see figure 2.2), for example in the footer, could be advantageous as the visitor will naturally end up at it when the page's content has been consumed. It could consequently become a way for her to move along to the next place. Using it as the sole method of exposing the site's navigation might be troublesome even if it is floating as users does not seem to be looking down for a menu (Dawn Shaikh and Lenz, 2006). Rather, it can be a good complement to a primary navigation as many people expect some form of links (especially a link to the *about* page) in the site's footer (Dawn Shaikh and Lenz, 2006).

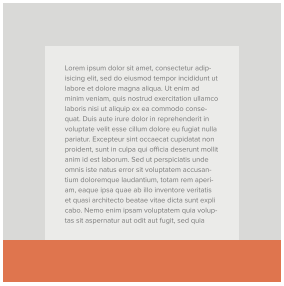


Figure 2.2: Menu at the bottom of the site.

The most common location where users looked for a menu system of links to other parts of the site was on the left hand side (see figure 2.3) according to a study by Dawn Shaikh and Lenz (2006) as well as the F-shaped reading pattern (Nielsen, 2006). One possible flaw with these studies is that they were performed in an English speaking country, or in other words with the possibility of a bias towards having the menu on the left rather than the right might exist due to the reading direction of left to right. Krug (2006) mentions in his book *Don't make me think* that people who are used to a left to right direction expect the site's logo to be at the top left while those used to a right to left orientation expects it at the top right. This is also connected to the language of the actual site, i.e. if it is in English people will most likely expect it to be to the left and if it is in Hebrew they will likely expect the logo to the right. The same logic could very well also apply to the location of a menu. Other than this having the menu on the left or right hand side (see figure 2.4) of the content provides the same advantages and drawbacks.

In terms of advantages, a left or right hand side menu can have the benefit of not encroaching as much on the content area as other positions when the number of items in it grow. Although long items might be problematic, and require encroaching on the content area or need wrapping, it is manageable. As it would then not be any major problem to always have the menu visible, showing current page and the sections it belongs to, it could serve to show where the user is, what is available and where to go next. On the other hand a possible problem with having a menu here is that the site might become unbalanced visually. More so if the menu is floating a scrollbar might be needed quite soon for certain sizes of the user's browser window.

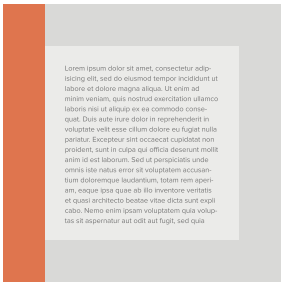


Figure 2.3: Menu on the left hand side of the site.

It might be possible to combine several menus to achieve a greater navigation system than with only one menu (Krug, 2006). Care must in such case be taken that they do not compete for the showing the same information as it could become confusing with multiple states as the user could have to ask herself which menu to use in any given circumstance. The latter could lead to a situation where the visitor is more focused on understanding how to use the site than understanding the content or focusing on her task (Nielsen and Loranger, 2006).

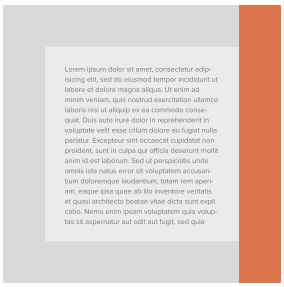


Figure 2.4: Menu on the right hand side of the site.

Much like with a left or right hand side menu, visitors expect there to be a site identifier (most often in the form of a logo) at the top left corner (Dawn Shaikh and Lenz, 2006). This given that the site's language is left to right oriented otherwise the identifier is expected in the top right corner (Krug, 2006). This element should not only exist from a pure branding perspective, to show that the site and its content is provided by a specific party, but also to instill a feeling safety

and security. Furthermore in order for the visitor to know where she is but also provide a way back to a safe location, i.e. the home page (Krug, 2006). It is also a good idea to supplement with a way to get back to the home page by an explicit link somewhere in the main navigation area.

As mentioned above, all layouts and positions will have to deal with when one or more dimension gets too small to contain the entire menu without having to scroll. While people are used to and can handle scrolling vertically, scrolling horizontally is decidedly more difficult and disliked (Nielsen and Loranger, 2006). Consequently navigation system which depend on the width of the window can be more sensitive to a shrinking window. Yet all layouts will eventually get to a point where it can not be contained. As such it is important to think of the site's and navigation system's responsiveness to the container size.

2.6.2 Search

Besides using the main site navigation and links in the content (Krug, 2006) users also, to a large extent, use site search to navigate to find the information they are looking for (Nielsen and Loranger, 2006). With persons in the age span of 18 through 31 more or less regard search as a requirement (Djamasbi, Siegel and Tullis, 2010). As such it is important for the site to have an integrated search functionality which is exposed so that the user can find it easily and then also refine her search if the results where not the sought. The content of a developer site might be made up of specialized data which might not be suited for a full text parser. Instead being provided in its raw form, adjusting how the search functionality can accommodate for this and provide a more specialized experience could be interesting. Unlike what generic search engines are able to.

The search functionality should be exposed through two means, first of all as a search box which is available from the same location on all pages of the site (Nielsen, 2001) as well as from a dedicated search page. The latter, dedicated search result page should also offer the user ways of refining the search in ways suitable to the content and resources provided by the site. One should avoid using terminology such as "advanced search" as it does not actually communicate what it provides and serves to confuse the user or make her doubtful (Nielsen, 2001). Continuing with the result page, research by Cutrell and Guan (2007) tells us that the results should be ordered by rank as the user expects this and trusts the search engine to provide the correct resource first. It should also have a title, the URL (uniform resource locator) of the result and a excerpt from the match. Furthermore Cutrell and Guan discuss the importance of their order as well as the length of the excerpt where a longer snippet of text is more appropriate for information tasks and shorter snippets when the search functionality is used as a navigational tool. A suggestion made in the article by the aforementioned authors was to place the URL in between the title and excerpt as it would "guarantee that as users scan the results the would always see the URL before the snippet". This in turn would make the page work

better for users using it as navigational tool while at the same time, given adequate styling, not interfering with users' looking for information.

An equally important factor as how to display the search results is what to show when no results are found. If possible the search engine should always try its hardest to return at least one result, say for example if a different spelling or synonym would produce a set of results that should be display while also informing the user of how and why the search query was interpreted differently. The user should never be asked to perform an action to show some results, this is something which the system should provide automatically. In the case where no results or possible results can be found even after trying ones hardest other ways for the user to continue should be provided (Travis, 2009). Such as;

- An escape hatch back to the home page so that the visitor can start over from a safe location.
- Relevant links to for example the site's different sections, or if a specific section can be identified to pages of interest.

Returning to the site wide search box, the field should most likely be located at either the top left or top right corner (Dawn Shaikh and Lenz, 2006) of the site as these represent the two places where the user will look first. There is a higher tendency to look at the top right corner first. This behavior and thereby suggested placement is also supported by the F-shaped scanning pattern found in by Nielsen (2006). The box should also be sized so that it can accommodate the most common queries. Such data can be gathered via site analytics (discussed in section 2.1.2) if available. Otherwise using making sure that at least 27 characters fits is a good measure (Nielsen and Loranger, 2006). Such width can on a website be achieved by setting the search fields width to *27 em* as one (1) *em* represent the width of the current point size (e.g. a point size of 16 points leads to a width of 16 points). Furthermore the site wide search field could provide smart auto-suggestions when the user inputs text. It is benefitial to allow the user to directly select the correct answer without having to visit the intermediary search results page.

2.7 Home page

The home page is the single page through which the largest amount of users enters the site via (see the analytics in section 4.1.3). Special care needs to be taken when designing it. It is important to emphasize that the home page is not the single page where everyone enters the site and it should not be the required way either (Nielsen and Loranger, 2006). Rather it is the page which on its own sees the most traffic entering the site. Therefore the home page is a rather special and important page. There are generally two different forms a home page of a developer site can take on, either showcasing and providing the entire site's content or as a gateway to the rest of the site. In the latter case where it serves as a gateway it should also be the only page on the site with that purpose (Nielsen, 2004). Looking at the two cases a bit more closely reveals that both types actually act

as a funnel for the user to get to some information or resource. Be that on a different page or location on the current page.

In both cases there are quite a few shared factors to keep in mind. The first factor to keep in mind is how how much time users spend on the home page. As previously mentioned (in section 2.1.2.1), users with a low experience with the web views the home page for 35 seconds on average Nielsen and Loranger (2006). Furthermore, the visitor will spend most of that time trying to figure out where to go next. Thusly the text describing the purpose of the site and what is being offered, as well as why the visitor should bother, must be tremendously short and concise (Nielsen and Loranger, 2006; Krug, 2006). Nielsen and Loranger (2006) even provides a specific recommendation of only using ten to twenty words. Also, the more text a page contains the less visually appealing it could be perceived (Tullis and Tullis, 2007). Yet it is important to communicate what the site is all about as well as what is offered and why the user should use what is provided (Nielsen and Tahir, 2001; Krug, 2006) instead of, for example, a competitor's similar offering.

More so, people using the web do not actually read the text on pages (Krug, 2006; Nielsen and Loranger, 2006) but rather scan for information and actionable elements that matches what they are looking for (Krug, 2006). The scanning usually follows an F-shaped pattern (Nielsen, 2006) where the users starts out by scanning from the top in a horizontal direction (see figure figure 2.5). Then moving downwards until the next element group where she will once again scan in a horizontal direction. This is repeated a few times where each iteration leads to a shorter distance scanned in the horizontal direction. Lastly Nielsen (2006) describes how the user finishes off with a vertical scan of the left hand side. Nielsen does not divulge whether where the participants are from which could obviously skew the results depending on their preferred reading direction. Albeit that this pattern means that the page's text should be optimized for quick understanding, perhaps even aimed at a person in eighth grade (Nielsen and Loranger, 2006), the start of each paragraph or element group should contain the most vital information. Besides being found more visually pleasing (Cober, Brown and Levy, 2004), using images can also change how the site's visitors scan the given page. As shown in an eye tracking study by Djamasbi et al. (2010) visitors will focus their attention on faces, if present, as well as the content which these faces seem to be looking at.

Since the visitor only spends a short amount of time one the page before deciding whether it is of any interest (Nielsen and Loranger, 2006) the placement of content must also be considered. Only a small fraction of users scroll the page (23 % on their first visit and then less on subsequent visits, Nielsen and Loranger 2006) and those who do spend less attention to the content below the fold than the content above it (Nielsen, 2010)[7]. Consequently the home page's most important content must be place above the fold for the most common resolutions.

[7]The *fold* is the the lowest visible point on the website in the user's browser window, requiring no scrolling or other action for it to be visible. In other words, one screenful of content.

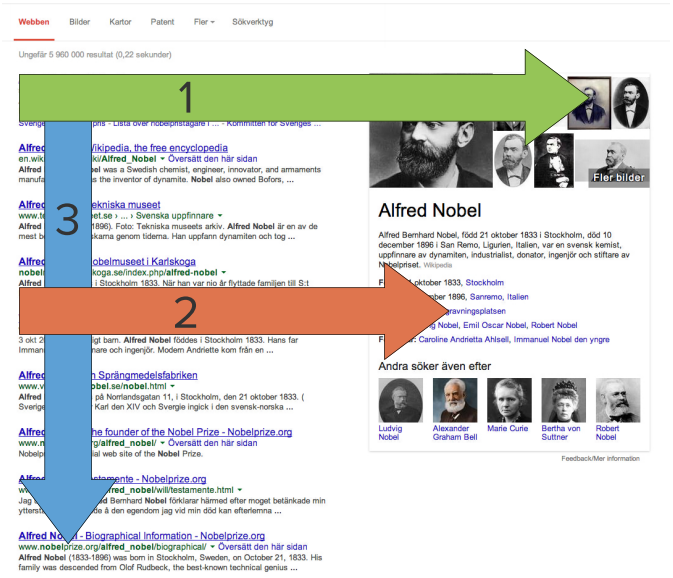


Figure 2.5: F-shaped eye scanning movement.

One of the most important goals of the home page is, as discussed in a paragraph above, to direct the user to where she wants to go next. One method of accomplishing this is to provide clear paths for the user. When creating these paths the experience levels of the targeted users needs to be taken into account as they might be looking for very different types of content and ways to get to it. Furthermore different persons need different ways of arriving to the same set of resources, as long as they are not confused to be the same type of element (i.e. two elements both being perceived as the main navigation will cause confusion). It can consequently be a favorable quality if there is for example a main navigation as well as specific entry points in the content area of the page and a search box. When knowing what they are looking for, some people prefer to browse the main navigation looking for categories which could contain it while others will look for a search box immediately and use that (Krug, 2006). Uncertain or new users might instead want to find information answering the questions of what to use, if it solves their problem or how to use it.

As some users start out using any search functionality available on the site and as others fall back on it if they can not find it by manually browsing the site (Krug, 2006) it would seem important to include easy access to it on the home page. It turns out that this is also the case (Nielsen, 2001), especially for younger (age 18 through 31) visitors where it seems to be more or less of a requirement (Djamasbi et al., 2010).

We summarize a number of items which need to be present or answered on the home page:

- Where the users have ended up, what is this site and who are the people or organization behind the website? (Nielsen and Loranger, 2006; Nielsen and Tahir,

2001; Krug, 2006)

- Why should the user bother using the tools or services provided instead of someones else's similar offerings? (Nielsen and Loranger, 2006; Nielsen and Tahir, 2001; Krug, 2006)
- A way, for any visitors of the target groups, to get to the next place. (Nielsen and Loranger, 2006; Nielsen and Tahir, 2001; Krug, 2006).
- A search box. (Nielsen and Loranger, 2006; Djamasbi et al., 2010).

Another aspect which can help users understand what is offered is to show examples such as a code snippets showing how to use the library, how to use an API endpoint and the data returned or something else which suits the offering. Each example should be accompanied by a link to a detailed explanation of it as well as more similar examples (Nielsen and Tahir, 2001). The examples should, like the copy on the page, be kept short and concise. Therefore whether examples will fit the home page or not largely depends on the amount of offerings, and size and complexity of them. A large or complex library might be hard to demonstrate in a short example while a simple library or function might only require very few lines. On the other hand, if what is being offered consists of several simple libraries or functions the amount of examples needed to demonstrate all of them might be to many. What is clear though is that any such examples have a lower priority than the methods for the user to understand where she has ended up and how to get to the next place.

As the primary difference between the previously discussion types of home pages, all content on the page or a gateway to other pages, is the amount of actual content the argument of which type to chose also boils down to content size. When selecting one of them it should be considered whether the amount of content fits on one page or needs to be split up over several. This might not just be because of length but also diversity of the covered topics. Furthermore users prefer scrolling over paging through several pages but do not give content below the fold as much attention (Nielsen, 2010). Depending on the length of the content it might be advantageous to provide it all on one page if it can get to the point quickly (Nielsen, 2010). On the other hand a long page might deter a visitor or make it hard for the user to enter the site via a search engine and end up at the location of the sought resource without any further actions. Lastly any such choices will likely need testing and comparison given the specific nature of each developer site.

3

Method

The methods used in this project were very much focused on user research in order to generalize characteristics to design for. It also consisted of data analysis of different kinds. The following chapter will in detail describe what methods were used toward a design driven result.

3.1 Design based research

We take a design based approach to our research. What this means is that by producing an actual design we will on the way there identify underlying results. A design based resarch method can be described as a set of procedures. Its goal is producing theories, products and practices and during the process point out results (Barab and Squire, 2004). This form of approach opens up for continuous iteration cycles of analysis and redesign (Dede, Nelson, Ketelhut, Clarke and Bowman, 2004). According to a presentation from DC Design Talks (Jackson Fox, 2008) there are many names for the same approach but it all boils down to the same concept; having a theory, figuring out the data facts and understanding your user, designing for her reality and her goals and revising requirements and design choices iteratively. We take on a process as described in figure 3.1. It consists of understanding the user in the research phase to then ground design decisions in the modeling phase. When having modeled goals we form requirements which lead to a framework. Design driven research also implicates iterating between producing the framework and tweaking the requirements as seen in figure 3.1.

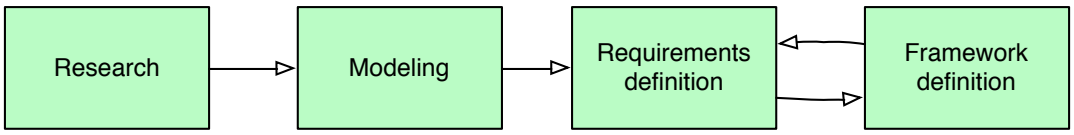


Figure 3.1: The Design Driven approach to product design.

3.2 Data collection

In order to root our assumptions we conducted several data collection processes. Interviews with stakeholders and users, observation of users, literature review, site statistics analysis and competitive analysis lay the foundation of our research material.

3.2.1 Observation

In order to gain quantiative data we performed observational studies by doing a contextual inquiry (Cooper et al., 2007) on about 100 students at a Way Out West Hackathon (Mager, 2013). A contextual inquiry is defined by Cooper as a sort of immersive observation technique which allows for meeting users in their natural environment with real occupations. It took form in over-the-shoulder observations during two full days. We took a passive viewer role where we stood behind them and noted down how they used developer sites in balance to how they coded. With about 20 of these participants we then approached them after the observations and asked them questions about their backround, habits, preferences and clarification on questionable observations we made. The method consists of having a so called "master-apprentice model" approach where you as interviewer act as an uninformed apprentice and let the user act as master. Balancing the observer and participant role is important in order to achieve both collaborative discovery and passive intake of information. Further it requires a balance when it comes to directing the user and letting her speak freely. We don't want to force the users down a particular path but still obtain answers to our questions. This was prepared by loosely forming interview topics instead of concrete interview questions, having an open and curious mind and allowing the user to fill the void (Travers, 2013). By stepping into the user's natural environment we opened up for more honest ethnographic impressions than we would have gotten in a sterile interview setup. Finally, and most challenging, it was up to us as designers to filter out what we perceive as important information, then these observations where transcribed and analyzed.

3.2.2 Interviews

We studied Travers' book (Travers, 2013) on successful interviewing. Further we used his five steps of interviewing; recruiting, preparing, conducting, documenting and synthesising. We conducted interviews with recruits from both the inside and the outside. This means that both external professional developers and employees at Spotify who work with QA/testing of third party apps. The Spotify employees daily confront products made with the help of the Spotify developer website as well as use it frequently, therefore they could give valuable insights into their needs. Furthermore, insiders are easily accessible and can argue for factors which outsider cannot; business values, user commitment, etc. When preparing for interviews not too much was scripted. Instead question areas were

defined in order to let them tell their own stories. Using the 5-whys (Sondalini) method was used to dig deeper into causes of opinions. An interview structure was set up as following: a prologue where we explained our purpose, small talk where we make the interviewee get accustomed to free talking, objective where we discuss our defined topics and finally a wrap up where we thank the interviewee and what how we will proceed with the information they've given us. Transparency and comfort was essential for a positive interview setting.

3.2.3 Competitive analysis

In order to compare the performance of Spotify's developer site against others' we measured UX heuristics (Bowles and Box, 2011) on a set of different developer sites, a so called competitive analysis. The ten sites we measured were chosen based on our own preferences, a poll sent out internally to a selected group of developers at Spotify as well as based on general popularity or community size. Although each site was not a direct competitor to Spotify, with regards to the service they offer, benchmarking against other well-designed web sites is an effective source of inspiration. This method would also allow us to get a general feel and understanding of what others succeed well in and not. By digging deep into the usability of a lot of developer sites, we hoped to get good insights in some of the factors that lead to a great developer website. In order to conduct a thorgough competitor analysis we used a method developed by Dr. Davis Travis called "247 Web usability guidelines" (Travis, 2009) where 247 parameters in nine different topics are graded from -1 to 1, or skipped if irrelevant, in an excel sheet for each site. By filling out scores for each question the overall total score is plotted out in form of a radar plot. Although a competitive analysis is merely one of several methods of research, it can nonetheless be a good metric for improvement; the more round and filled, the better. But as Bowles and Box (Bowles and Box, 2011) put it *"Don't get hung up on competitors. If you're always chasing their tails, how will you ever get ahead?"*

3.2.4 Site analytics

We used analytics data gathered using *Google Analytics* from Spotify's current developer site (`developer.spotify.com`). The method was primarily chosen as it provides a way of gather a much larger datasets than possible, from a time and resource perspective, with direct interviews and observations. This data can once analysed be used to validate and strengthen our other research, such as the previously mentioned interviews and observations (see section 3.2.2 for those), on user behavior and provide design constraints. Design constraints both in terms of technical limitations of the target group's devices but also based on their own behavior. The analytics data was also used to point out possible discrepancies in our research data which might have needed addressing.

Although website analytics make it quite trivial[1] to gather these large datasets its primary drawback was the possibility of a time consuming analysis phase Hasan et al.. Time consuming for the very same reason why data gathering this way was helpful, the possibly large dataset, as well as the many variables that could be defined.

As stated we utilized the analytics service *Google Analytics* which provides both both client side and server side like data. The reason why we used said service instead of another tool or service was primarily because it was the service *Spotify* had enabled for the site and it had thus been running for quite some time already. Google Analytics could therefore provide us a larger data set for a period much longer than if we would have enabled some other service when we began looking at the problem. Google's service also provided most of the types of data thought to be of interest for the study as outlined in section 2.1.2 (see Google Inc. 2013*b* for a reference of what was available).

3.3 Personas

Based on data collected in interviews and observations we created a set of personas. We had two primary methods for inspiration when creating the personas empowering our design process, namely:

- *Ten steps to personas* (Snitker & Co, n.d.) based on the phd dissertation "Engaging Personas and Narrative Scenarios" Nielsen
- *Creating personas based on the concept of behavioral parameters* (Cooper et al., 2007; Goodwin, 2009)

The first method is based on a process which consists of a set of ten steps. It based on buidling a hypothesis, costructing material and verifying against the data that the hypothesis is correct. The steps are as follows:

1. Finding the users
2. Building a Hypothesis
3. Verificiation
4. Finding Patterns
5. Constructing Personas
6. Defining Situations
7. Validation and Buy-ins
8. Dissemination of Knowledge

[1] Most often by just adding a short code-snippet to ones website pages (see for example the help page "Set up the web tracking code - Analytics Help".

9. Creating Scenarios
10. Ongoing Development

The latter method is based on the principle of identifying keywords from interview notes which are then written down in the margins of the transcripts. The keywords should describe the topic of what they were talking about (i.e. mental models, goals, attitudes, skills, etc.). These then contribute to forming behavioral and demographic variables which are visualized on a horizontal line. The interviewees results are plotted according to level, low to the left and high to the right. When having done this one can begin to identify patterns which can then be converted into characterizations for a persona. After having grouped data into patterns, we identified distictions between them which then lead to futher details to our fictive personas. The process described by these methods can be divided into these steps:

1. Divide interviewees by role
2. Identify behavioral and demographic variables
3. Map interviewees to variables
4. Identify patterns
5. Define goals
6. Clarify distinctions and add detail
7. Fill in other persona types as needed
8. Group and prioritize user personas
9. Develop the narrative and other communication

Our process was a kind of pick-and-choose mixture of these methods. Since they somewhat resembled each other we could merge them into one method that suited our design process.

1. Finding and defining user roles

Based on collected data from interviews we developed a theory about who our users might be. At this stage *who* they are is defined by *what* they do. Task-orientation and job titles are thus more important than life values and characteristics. This is in order to make it easier to do a a first general draft of our target group without getting too detailed. Narrowing the scope and increasing level of detail should instead be done successively throughout the steps. The following possible groupings of developer site users were defined:

- Professional developers (Either experienced or inexperienced with the platform)
- Novice developers

- Designers (Interaction or graphics designers)
- Testers
- Business developers or managers
- Consumers

Furthermore we made a difference between whether these were internal or external users since if affects level of previous knowledge but also information access. Internal roles, meaning employed by Spotify, have more knowledge about the systems and tools provided and can also affect what is displayed externally. We also presumed that they have better opportunities of getting support than external users.

2. Prioritization

Observational studies and reserach limitation led to selecting the following subgroup of users:

- Professional developers (external)
- Novice developers (external)
- Testers (internal)

At this stage we have defined target roles.

3. Identifying paramaters

During a brainstorming session (Net, 2006) we developed a set of character parameters based on the data transcription keywords. We asked ourselves what it was that separated the interviewees. We found differences such as resources (time, money), experience (both platform and general programming), background, goals and patience. Then we identified topics that the transcripts had in common. Combining these we made the assessment that the following behavioral parameters were relevant to our study:

- Programming experience
- Leisure vs Business
- Time pressure
- Platform experience
- Collaboration level
- Resources

These parameters are visualized along a horizontal scale for each one as seen in figure 3.2.

Figure 3.2: Behavioral parameter scale

At this stage we have defined character parameters to map the roles against.

4. Mapping interviewees to parameters

Each interviewee's result was plotted out on paper according to their level, low to the left and high to the right. After this, we combined their individual results into an overall conclusion of persona characterizations. The goal of visualizing the data in this manner is to be able to compare each interviewee agains the others according to each variable. Precision it not the key but rather the overall inclination to one side or the middle, such as in a Likert (Preece, Rogers and Sharp, 2002) scale. By mapping out data (as can be seen in figure 4.15) it became easier to see patterns in behavior. The point is to get clusterings in different areas of the scale. If one parameter gets all the data gathered in the same point, it will not be useful for personas since we are looking for different patterns that will distinguish themselves.

At this stage we have defined connection between our roles and character parameters.

5. Finding patterns and verification

We faced ourselves with a set of parameters with data along the lines. Finding patterns amongst these are the first step to actually seeing the basis of personas take form. Making sense of the data can be done by circling in interviewee whose results frequently appear next together.

When having idientified the clusters (as can be seen in figure 4.16) it was important to evaluate the validity of them by checking if the initial groupings still held. We identified certain other groups which might also be important when designing a developer site, such as a salesperson or PR agent who turns to it in order to find information about value proposition and reasons for investing. Although they are important when designing they whole experience, they are note equally important to our study (see *Delimitations*).

At this stage we hav defined a set of patterns with several characteristics in common.

6. User goal hypothesis

Now that we have a basic understanding of the different types of charecteristics that will form our personas we set out to define *what* they want to do and let this drive our design decisions. This is formally known as Goal-Based Design (Goodwin, 2009). By knowing what they are like we can easier understand *how* they would go about in order to reach those goals.

We start off by defining some high-level contexts from which these end goals could be derived.

- Creating a hobby app
- Reviewing a third-party app
- Creating a company app with Spotify
- Participating in a hack competition

At this stage we have a set of goals for each persona.

7. Persona compilation

Next up followed verification of the hypothesis. *Do these match our previous findings? Does it seem like reasonable goals for the persona types we have defined?* We had encountered several people who matched these groupings and now needed to verify that

they aligned and that our data would support future design choices. We did this by further adding detail and cross-checking with our data.

We set out with three personas in mind:

- The professional developer with 10 years experience
- The semi-experienced computer science student developer
- The semi-tech app tester at Spotify

For each of the personas we chose to present different levels of detailed information; short information about their personality, the context in which they are in, longer detailed life description, quotes and what they would need from a devsite. The first two are to give the designer a fast grip on their core quialities and in what situation we can expect to find them. The longer detailed information is provided to give a deeper feeling for the persona in order to achieve qualia. This text is not intended to read again and again, thus it would contain less vital information and be presented less prominent. Quotes are short and concise and can be very representative of the person since we can image him/her actually uttering these words. We therefore we prioritize these giving them a lot of space. Finally, we provide their needs from a developer page. The aim is to formualte requirements based on these personas' needs. The risk with writing out specific needs is that designers will disregard all other information and just pick these right off instead. However, since we are aware of these risks and in the position of being both researcher and designer, we chose to retain this information.

Although there is some critique against using illustations for personas (Thompson, 2009) we chose to do this instead of photographs because it gave us the freedom of defining body type and style ourselves. Another advantage is time efficiency as we avioded spending time on finding a suitable photograph or person to photograph.

At this stage we have three personas.

8. User scenarios

Without situations or problems to solve, personas are nothing. However in a scenario they become useful. From there we can. From the contexts and with the personas in mind the following potential user scenarios were defined.

- Get sample code
- Understand how to get started
- Read API documentation
- Find inspiration for what can be done
- Understand rules and regulations for app dev

- Manage developer accounts
- Get support when stuck
- Search for error message

At this stage we have three personas and possible pertaining scenarios to place them in.

4

Results

We set out to research the area of developer pages and how these could be improved for the developers. Based on general findings and Spotify's developer page as a reference case we adopted a design driven approach we developed a new design for developer.spotify.com. The design derives from a set of recommendations we identified that should be applicable to a wide range of developer sites. In this chapter we present both the results from our benchmarking analysis, user studies and our final prototype.

4.1 Review of developer sites

In this section the results from the competitive review using the 247 method by Travis (2009) are presented as well as analysis of the scores each site was awarded. The results attained by going through the web analytics from Spotify's developer site at the time of the research is then presented. The former results can help build an idea of what other parties are doing and what they are doing well at such that inspiration and learnings can be drawn from it. While the results from the latter study can help identify behaviors and needs the visitors of the site had at the timeframe of the analytics data.

4.1.1 Competitive analysis

A review of ten (10) different developer sites, including Spotify's site, was performed as discussed in the method section regarding competitive analysis (3.2.3). As the user studies (see section 4.2.1) and literature (see section 2.6.2) had shown the importance of search the original 247 method spreadsheet by Travis (2009) was modified such that if a site had no search it would get a worse score. Something which was not present in the original version. The review was carried out between 2013-07-25 and 2013-08-14. The method provides a way to review eight separate, although more or less interconnected. parts of a site namely;

- Home Page

- Task Orientation
- Navigation & IA
- Trust & Credibility
- Writing & Content Quality
- Page Layout & Visual Design
- Search
- Help, Feedback & Error Tolerance

As mentioned in the method section ten developer sites were picked for the study. Below you will find a list of all the sites whose developer site was reviewed ranked by the score it received (higher to lower, where position one (1) represent the best scored site) in the review using the 247 method.

1. Android
2. Twitter
3. Facebook
4. Xamarin
5. App.net
6. Apple
7. Dropbox
8. Microsoft
9. Spotify
10. Soundcloud

Each individual review is gone through and analysed below, in an aplhabetical order, along with the score it recieved.

4.1.1.1 *Android's* developer site

Review score: 78 %

The Android developer site excels in trust and credibility with the score of 95 %. This is partly due to the facility of quickly accessing updated company information, clear branding and free of errors. Although the home page aided in a good score in trustability due to a confidence inspiring design, it lacks in usability. The main reason for this is because there is no clear indication of which paths the user may take. Should the user not be entirely sure of what she is searching for, then the starting page does not provide

a natural flow and exploration encouragement. The major options are clearly displayed but this might not suffice for an inexperienced user. Despite this, the overall navigation is clear and it is evident how to get from one place to another and what actions do.

The search functionality is initially very nice and gives the user many input suggestions and recommendations in a well visualized way. The scoring is lowered due to the difficulty in browsing the search results and suggestion of similar topics and simple search aid (such as spelling correction, advanced options and similarity search result).

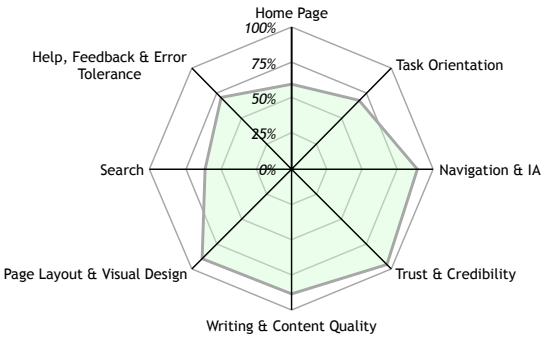


Figure 4.1: Radar chart over the Android developer site's score in the eight different areas.

The entire page is nicely linked together, from any stage it is easy to navigate to another place and then get back to the starting point. Many internal links between e.g. design docs for buttons and the code guide reference. Furthermore the visual design is really aesthetically pleasing and easy on the eye.

Overall the Android developer site received a score of 78 % putting it at the very top of the reviewed sites. Although it ended up as the "number one" site compared to the other sites that in itself does not imply that the site is in any way perfect. There are quite a few areas where it could be improved or perhaps should be designed in a different way.

4.1.1.2 *App.net's* developer site

Review score: 74 %[1]

App.net's developer site received an overall rating of 74 % where the major benefactors where the writing and content quality, the trust and credibility conveyed as well as the navigation and information architecture. Although the site scored well on the mentioned areas and relatively good on most of the remaining areas, the complete lack of search severely brought the score down. With the unmodified version of the 247 method the site would have got earned a score of 85 %, putting it squarely at the top of the review. As our version of the review method gives a more prominent penalty for not having a critical feature, such as search, this brought the grade down and place App.net' developer site on fifth place.

[1]Using the original (by us not modified) method App.net's developer site got a score of 85 %. The reason for diminished score is due to missing search functionality.

When it comes to areas the site performs well in the information architecture comes out as a good source of inspiration for further work. A key reason for this is that the site has been organized in a broad manner instead of a deep one. Meaning that the navigation structure exposes a lot of different pages but none of them leads to another page which in turn leads to another page (and so on) that is not available in a direct route from the home page. Using the broad pattern means that the user can much more easily create a mental model of the site which matches the actual structure and thusly find information and also find it again at a later stage.

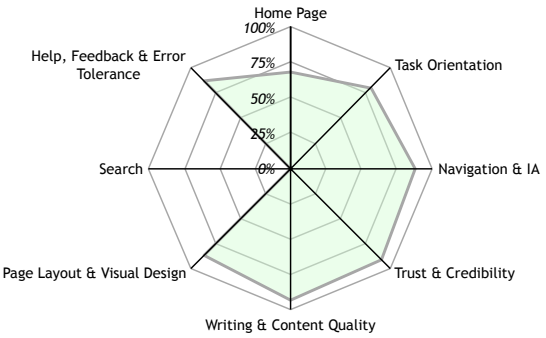


Figure 4.2: Radar chart over the App.net developer site's score in the eight different areas.

Another place where the App.net developer site performed well was the layout and visual design of pages. Here the information architecture and menu system also helps to inform the reader of where she can go next, yet it is not as prominent and obvious as on for example the Android developer site. On the other hand the content on each page is well structured and have a clear definition of what that specific page deals with and then more detailed information follows, sectioned under concise and easy to understand headings. In turn the text is well linked up with content in other parts of the site allowing the user to dig deeper in the information while still being able to keep track of where she is and where she has been.

Lastly it should be mentioned that the home page of the developer site is not that much of a home page telling the user quickly what is offered and where to begin but rather a more inforamtion dense page with content directly on it. This might in one sense be good since the developer arriving on the site quickly can get direct information. Unfortunately the amount of text and link density could also serve to confuse the user as well as overloading her with too many options.

4.1.1.3 *Apple's* developer site

Review score: 74 %

Apple's developer site excels, like many of the other sites, when it comes to the trust and credibility conveyed to the reader but also at the page layout and visual design. Less so when it comes to the task orientation as well as navigation and information architecture which both scored quite mediocrely compared to the other sites with a score around 55 %. The remaining areas received a score between the top and bottom and averaging around 75 %.

When looking at each page individually at the Apple developer site the all seem very well designed with good use of typography and supporting assets such as images, tables, listings and so on. The text of the site has also been laid out such that line lengths are within an acceptable length. Buttons and elements which perform an action, like links, can easily be discerned from non-interactable elements. There is also no need for the user to "hunt" for the actions as they have been designed such that their affordance is apparent.

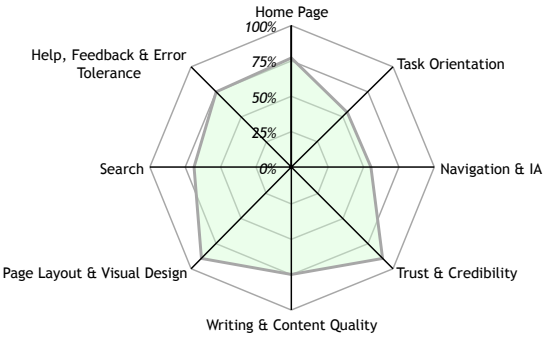


Figure 4.3: Radar chart over the Apple developer site's score in the eight different areas.

The major issue with the design and page layout on the site originates from its incoherence. If one starts to look at site more as a whole instead of the individual pages a set of different styles appear. Although pages which followed a specific style were coherent with other pages of that style there were major differences between the various styles which could possibly not just confuse the visitor as to what site she had arrived upon by clicking the previous link but also decreasing the trust and credibility instilled in the user.

A more worrying area where the site showed to be lacking is the navigation and information architecture as well as the task orientation as could be seen from its low scores in the two areas. The primary issue can be derived from the depth of the site navigation. Such deep navigation often leads to it being difficult for the user to create a mental model of the site. Due to the difficulty of creating a mental model it can also be strenuous to find ones way back to some information at a later stage. Many times the amount of clicks needed to perform a task of get some specific information greatly exceeded 5 clicks.

Unfortunately the search functionality of the site did not provide enough help to alleviate the the problems introduced by the navigation structure. Many times the user had to refine the search manually by using the browser's built-in page search functionality to actually find the correct position of the match which had been reported by the site's search functionality.

4.1.1.4 *Dropbox's* developer site

Review score: 73 %[2]

The Dropbox developer site does not score remarkably well in any one specific area but instead has a relatively high score throughout the entire review. With the exception of search which was missing completely from the site and thusly brought the score down

[2]Using the original (by us not modified) method Dropbox's developer site got a score of 83 %. The reason for diminished score is due to missing search functionality.

quite a bit. It did however have one of the better home pages and was quite good at task orientation throughout the site.

The home page is a good condensation of what is offered to external parties, and can be done with the service while also serving as a quick way to get to the actual content. The main focus of the home page is to lead the visitor to one of the different offerings, each accompanied by an image which provides a rough idea of what the product can be used for. The task orientation of the site is even more apparent due to the main navigation menu that shows all the options available. Even though Dropbox had several different APIs available to third parties the breadth, as opposed to depth, of the site makes it relatively easy to get around and know where one are.

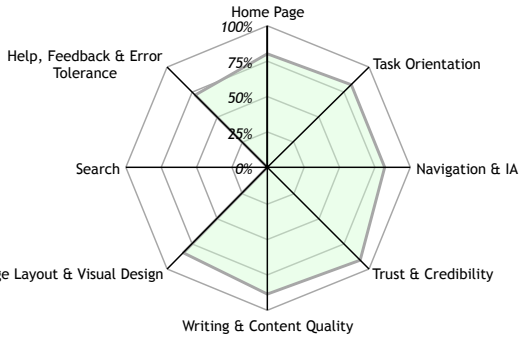


Figure 4.4: Radar chart over the Dropbox developer site's score in the eight different areas.

The only deviation from this were certain pages which lead the user away from the main navigation to a, similarly designed, page with its own top-level navigation. Furthermore these pages would open in a new tab or window even further exacerbating the experience. This along with the main navigation menu not being exactly organized as the content on the home page could lead to some confusion in the user.

4.1.1.5 *Facbook's* developer site

Review score: 78 %

In third place of our review comes Facebook. With a strong presence in the community world, thousands of third-party developers and a known brand among tech companies, their developer page is probably frequently visited and has a high average of daily views.

Developers.facebook.com partly achieve its high score thanks to good, valuable and well written content but more relevant to our study because of easy navigation as well as a clean visual design. The clean visuals of the dev site contribute to the direct positive impression and a design which does not confuse but serve as inspiration while aiding comprehension and not create confusion about e.g. elements. However, they lack somewhat in visual consistency throughout the pages.

The Facebook developer site manages to create an understandable and logic navigation on the page. Information is structured well and directly from the start page the user can get on the right path according to their preference. Providing this as early as in the landing page contributes to both scoring high on the home page, task orientation and navigation & information architecture.

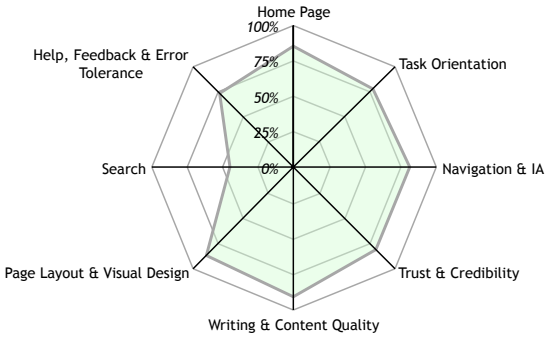


Figure 4.5: Radar chart over the Facebook developer site's score in the eight different areas.

Where this site lacks in usability is in their search implementation. Search results are not displayed in a clean and lucid manner. The search box is placed where it is in the normal Facebook interface, which can be both positive and negative. It contributes to alignment in design between products but might also confuse the user with the difference between dev site and facebook.com. So whether or not it is placed in a intuitive manner, is not entirely clear. Further, refining search and search aid is not supported well.

4.1.1.6 *Microsoft's* developer site

Review score: 64 %

When it comes to Microsoft's developer site, msdn, there is no doubt that the user gets a feel of trust and belief that the site is legit. With a graphical style which is in direct line with Microsoft and well written text, the site feels robust.

The problem of Msdn is its size. Because it is so extensive and has so many possible paths with unclear navigation patterns the site becomes confusing for the user. From the first start it is not clear which possible paths there are, probably because there are so many. There is so much content that each underlying platform has it's own style. When browsing the site, you easily get lost and a feel of having left the page because of the shift in layout and design when being redirected to a sub-platform's own home page. The Microsoft developer site is thus in fact a collection of many underlying developer sites which makes finding your correct path cumbersome.

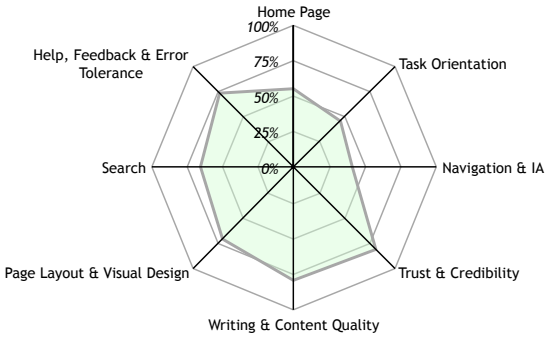


Figure 4.6: Radar chart over the Microsoft developer site's score in the eight different areas.

When the user finally reaches the sought out destination, msdn succeeds in presenting well written documentation that free from questions. Although it is outside our scope, msdn does actually succeed in convey a business investment message which motivates to indulge into their services.

4.1.1.7 *Soundcloud's* developer site

Review score: 59 %[3]

Soundclouds documentation is well written and the code snippets, inline widget examples and good linking stands behind augmenting the score somewhat.

Overall Soundcloud's page scores really bad. Navigating between tabs and sections is cumbersome and misleading. Stepping back does not always lead back and the user might get lost in the navigation. On top of this, there is no search functionality which further obstructs the user from reaching the sought out destination. Neither does Soundcloud help the user to get on the right path once they have gotten on the wrong track.

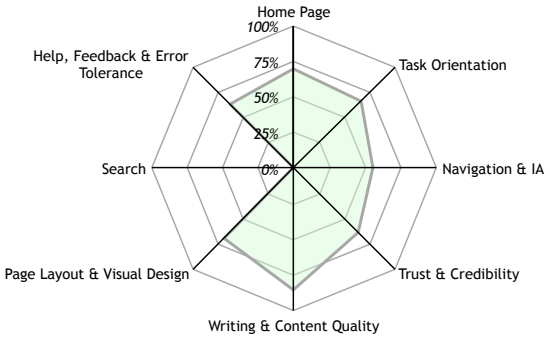


Figure 4.7: Radar chart over the Soundcloud developer site's score in the eight different areas.

In the case of the home page Soundcloud partly shows the possible paths presented to the user. However, there are several features (such as hidden affordance and malplaced feeds) that might disturb the visitor from finding the correct path immediately. Also, the home page gives no clear visual of the structure of how the web page is in fact constructed.

4.1.1.8 *Spotify's* developer site

Review score: 53 %[4]

The overall score for Spotify's developer site was the lowest of all the reviewed sites at 53 %. The primary reason for this is the lack of site search, a very low score in regards to navigation and information architecture. More so several other aspects also received a quite low or mediocre score.

[3]Using the original (by us not modified) method Soundcloud's developer site got a score of 68 %. The reason for diminished score is due to missing search functionality.

[4]Using the original (by us not modified) method Spotify's developer site got a score of 61 %. The reason for diminished score is due to missing search functionality.

The Home Page provides the user with three main paths. However, it is not entirely clear what the final product from these paths will be. Although we have favorized visuals which contribute to the feel of the page, in this case there is too much visual and too little actual valuable content.

When trying to navigate the page it is happens too often that the page shifts entirely in layout and design. These visual changes partly makes it hard for the visitor to know what site she is on and thusly decreasing the trust of what is made available through the site. Furthermore it also becomes troublesome to find ones way or orientate oneself as the canvas often changes. The navigation problem is also made worse by the fact that the different page layouts also have their own main navigation menu. Sometimes even trapping the user in a specific part of the site with no or very limited ability to get back to the rest of the developer site.

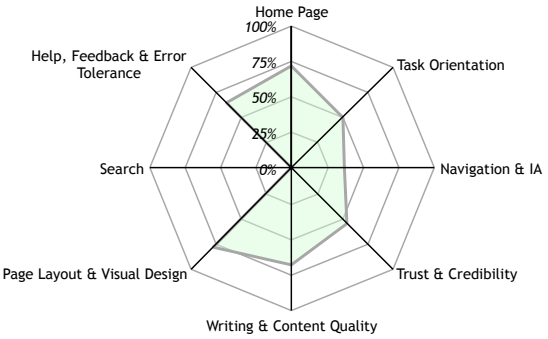


Figure 4.8: Radar chart over the Spotify developer site's score in the eight different areas.

Another factor which had an impact on the user's ability to create a mental model of the site's pages was the fact that the developer site is very deep instead of broad. Meaning that it takes quite a few clicks through a number of other pages to get to the content. To be able to get back to some form of information, given the fact that there was no search functionality present, the visitor would have to remember each step.

If looking at the page layout and visual design of each page individually, and thus disregarding the incoherence of the site, most pages perform relatively alright compared to the other sites reviewed.

4.1.1.9 *Twitter's* developer site

Review score: 78 %

Twitter's developer site performed overall second best compared to the rest of the reviewed sites even though it scored mediocrely when it comes to search and among the lowest in terms of navigation and information architecture. Their home page on the other hand was the one which received the highest score of all tested sites.

The only real issues with the home page was the order of the main menu navigation items which seem to have been ordered from least important, or useful, to most important. That is in the backwards order. On the other hand the home page did provide a good overview of, most of, the offered products and real content (such as examples and code

snippets) are available.

Twitter's search functionality was, as stated above, one of the areas which lowered the overall score. This was mostly because the search engine provides just the basic functionality and does not gracefully handle problems. As an example when it could not find any results the information presented states that the user has spelled the query incorrectly and suggests that the same user try out some different queries. Queries which the search engine could automatically have performed itself immediately.

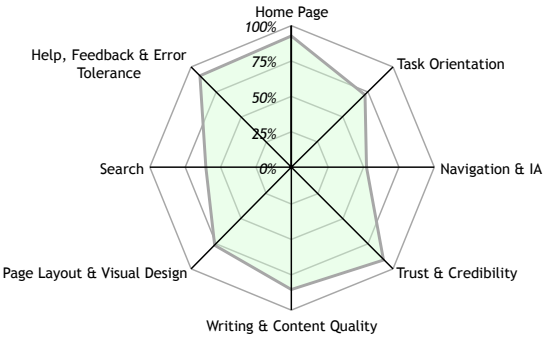


Figure 4.9: Radar chart over the Twitter developer site's score in the eight different areas.

Another problem with the site is its use of the primary brand color for sometimes too many elements. The problem here stemmed from the fact that the primary brand color is a quite saturated blue and when used with fine details (such as text and code examples) it can become rather hard to read. Furthermore, some pages which contained a large set of examples had a tendency of becoming visually cluttered. Mainly due to the fact that the example's came in a great deal of different shapes and sizes that were not separated enough resulting in a disordered page.

4.1.1.10 *Xamarin's* developer site

Review score: 76 %

What is good about the landing/home page for the dev site is that the user can quickly find samples, API's (application programming interface), tutorials, video instructions, etc. Since the developer page is on of the tabs of Xamarin's main web page the developer "site" gets immediate credibility. Furthermore there is contact information, social medias, copyright and terms conditions along with many other resourceful links in the footer that contributes to the trust. High content quality only further strengthens this.

When it comes to navigating the site, it is not entirely intiuitive how to get from one place to another. It becomes confusing when there are several levels of tabs, inconsistency between different parts of the page and differences in refining (sometimes a drop down, sometimes a sidebar, etc.).

Again, the search parameter is the one which diminishes the total score of the dev site. The auto-complete here is really nice and categorizes the search results in a clean way. From the drop down and auto completion suggestions, the user can quickly filter out irrelevant search results. However, their search engine handles empty queries really bad

as it interprets this as "undefined", in other words a search for null. This produced about 50 results when instead it should support the user and help improve the search query.

As far as the visual design goes, it is clear and understandable and gets a high score. Affordance and visual feedback is not misleading and it is pleasant to the eye. What the score does not fable is that is does not inspire or encourage creativity. Although there are graphics on a deeper level (when reading about a distinct topic) the whole look and feel of the page does not say "let's create things". However, the design is well aligned with Xamarin's graphical profile (partly because it is in fact a sub-part of the entire page).

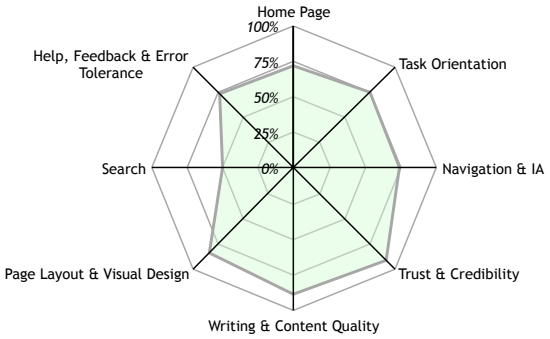


Figure 4.10: Radar chart over the Xamarin developer site's score in the eight different areas.

4.1.2 Information architecture

The current developer page has three main divisions; Technologies, Support and News. Three possible paths are centrally placed on the landing page (see figure figure 4.11); Spotify Apps, LibSpotify and Spotify Play Button. The choice of their placement could arguebly be based according to the the level of traffic that the elements drive and therefore be placed easily accessible to the visitor. In the upper tab menu, the home button and the upper left icon lead back to the home page which might indicate redundancy. Although all topics have subcategories, only the Technologies tab has a drop-down menu.

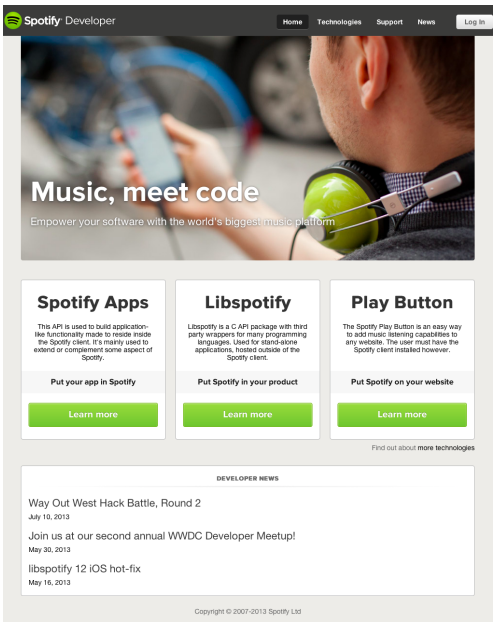


Figure 4.11: Screenshot of the landing page on developer.spotify.com

As can be seen in the diagram of the information architecture (see figure figure 4.12), there is a great inconsistency when it comes to navigation of subpages. Some lead to entirely other hosts such as Github or StackOverflow. Most of them are local but they vary greatly in design and layout. Site elements such as the API and guidelines are separated from the structure with a top menu bar and left content menu and instead percieved as entirely detached pages.

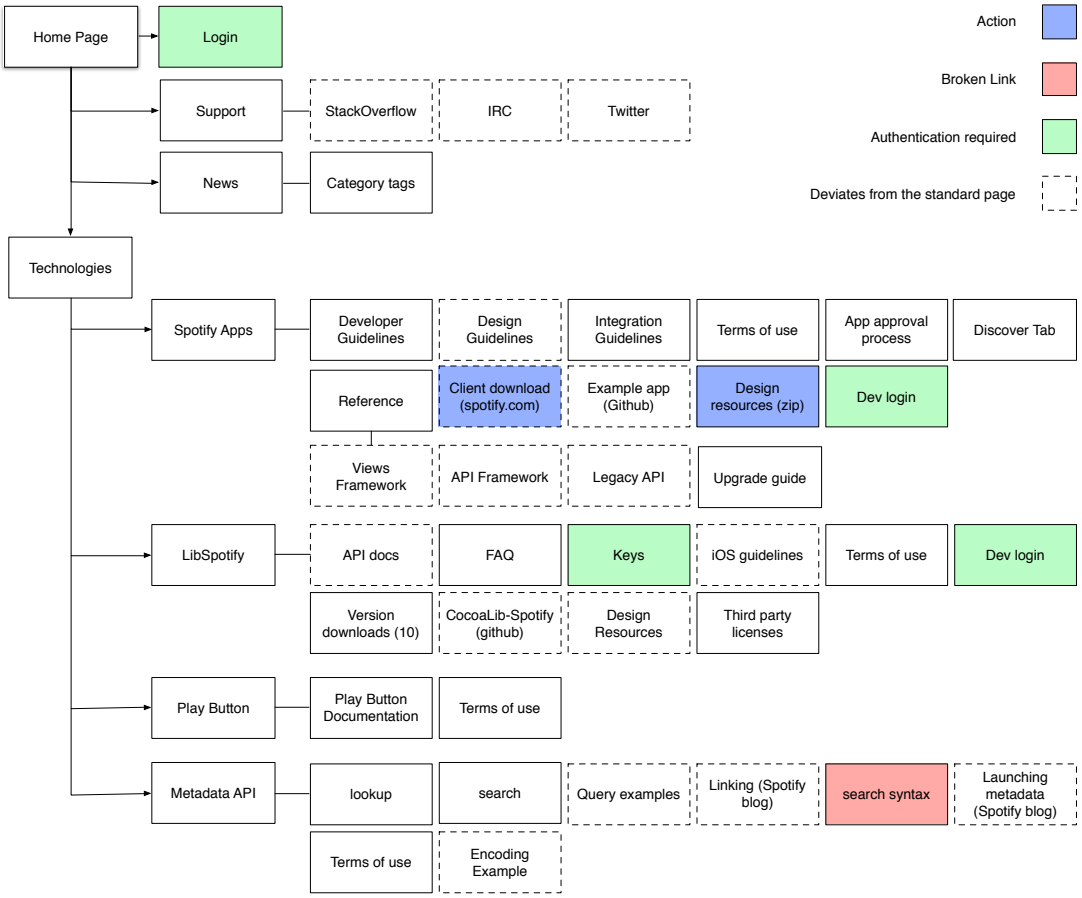


Figure 4.12: Diagram over the information architecture of developer.spotify.com

What cannot be seen in the information architecture diagram is the amount of content per page, but instead where subpages lead. Links that redirects the user to another location on the same page (eg. to a section further down) have been excluded from the figure. These are very frequent in the current design. Whether a wide and shallow or narrow and deep navigation is preferable for our target group is relevant to this research.

We ask ourselves if large groupings of content can and should be separated into smaller parts, what the purpose would be and if it might benefit browsing. Inconsistency with drop-down menus combined with static buttons might conflict with the users mental model. Isolation of subpages may seem bad at first glance but we aim to investigate whether there might be some advantage to this or if it breaks the flow of the user. The information architecture diagram will help us get the overall picture of how the content on a developer page is organized.

4.1.3 Spotify developer site analytics

As outlined in the theory section on the same subject, to get an idea of what the visitor of a developer site, their technical equipment and behavior, site analytics can be a great source of information and starting ground (Bowles and Box, 2011). We had a look at the analytics for Spotify's developer website for the period March 1st through August 31 of 2013, or in other words a period spanning six months. The specified date interval was chosen as it represent a large enough amount of data to provide good insights not swayed be single, small, events. This while also representing usage only during the period where the developer portal looked and acted as it did during our review of different developer sites (see section 4.1.1). The data was collected via the online analytics service *Google Analytics* throughout the dates mentioned previously.

Early on when looking at the analytics data we were able to discern two primary and separate groups of visitors of the Spotify developer site. Namely *developers* and, at first surprisingly, *consumers*. These two groups behaved completely different where the first one did confirm and behave quite closely to our expectations after interviews with, and observations of, developers (see section 4.2.1). The latter group of visitors on the other hand did not behave in the way we expected a developer to. Instead these (henceforth mentioned as) *consumers* almost exclusively visited a specific page (*Page A*) and then left the site without viewing any other information.

It was also noted that all but a few of the consumers entered the site on this specific *Page A* that they also left from. The traffic which flowed through the site's home page also gave more weight to our theory that these visitors were in fact not developers as the option to go to *Page A* was the least popular option. After reviewing the contents of said page it did become clear that it was in fact not intended for developers per se. On the contrary the page provided a way for consumers to integrate and play music on her personal website or blog using a pre-made widget that could be dropped in place without any development effort. It should also be noted that the page did most likely receive some visits from developers (integrating it into their own offerings) but these also seemed to behave more like the studied developers. Due to our delimitations in this thesis we did not consider the needs, behavior and requirements of these consumers directly.

We also noticed a difference in behavior which contradicted some literature and industry comments on the subject in regards to the bounce rate of the visitors as the visitor's flow on the site many times differed from that of a normal or e-commerce site. Bounce rate is often seen as a negative factor (Sculley et al., 2009) as you "normally" on, for example, e-commerce sites want the user enter on a given page A and then navigates the site and leaves on a different page B after having visited several other pages. Possibly also having the user performed an action such as a checkout or signing up for the service the site offers. When the user instead leaves the site from the same page as she entered, without visiting any other pages in between or reloading the entry page, it is counted as a bounce (Burby, Brown and Committee, 2007), which is also the definition the analytics provider used (Google Inc., 2013*a*).

On the other hand, in the case of a developer site we found that many developers' flow starts with an external search engine both in interviews with said group of people (section 4.2.1) and in the statistics of the Spotify developer site. As many as 32 % of the total number of visits (as can be seen in figure 4.13) to the site entered via a search engine (see appendix B). One could therefore argue that if a user enters the site on the page which contains all of the information she was looking for a bounce might not be such a bad thing. Especially for a site with the focus of providing its visitors a certain set of information as quickly as possible (as found out in the interviews (4.2.1) with developers).

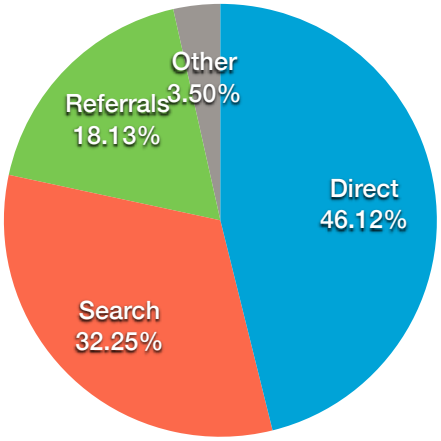


Figure 4.13: A chart showing the distribution of how visitors arrive at the developer site. One can see that 46 % arrive via direct traffic, 32 % via search traffic, 18 % via referrals and the remaining 4 % via various mediums.

4.1.3.1 Traffic from search engines

Search traffic did, as mentioned in the parent section (4.1.3), play a substantial role in driving traffic to the developer site and as discovered via interviews and observation of developers (section 4.2.1). As such we looked into the specifics of how visitors originating from search engines differ from the average user.

Users who entered the site via a search engine had a much lower bounce rate (48 %, almost 16 percentage points lower than the average) than users arriving to the site via other traffic in-vectors. In fact search had the lowest bounce rate of all in-vectors of the site during the measured period. The closest competitor was direct traffic with a bounce rate of 69 % which in turn was well over the average value. Search visitors also seem to be returning to the site more often (38 % as opposed to the site average of 24 %) than users arriving to the site via other means. These persons also had a higher site engagement overall with longer page visits and more pages visited.

Continuing with the site's user engagement it was found that most (69 % of all) visitors seem to drop off after zero to ten seconds after visiting on average 1.13 pages. Thus they did only account for 30 % of all page views. Visitors that stayed longer also on average seemed to view more pages. Perhaps the most remarkable insight was that the last 1 % visits which lasted for more than 1801 seconds also accounted for 9 % of all page views. That was more than the either of the 11-30 and 31-60 seconds groups where the former clocked in at 7 % of all visits and 7 % of all page views while came in at 6 % of the visits and 8 % of the page views. The visitors who stayed for a period of time inbetween 60 seconds and 1801 seconds resulted in the largest segment of page views (around 47 %).

4.1.3.2 The visitor's screen properties

Of notable mention the most common screen resolution was 400x800 points (these were almost exclusively *Windows Phone* devices) while on the desktop 1366x768 points was the most common. The resolutions 1920x1080, 1280x800 and 1440x900 points came in with at about the same amount of visits. It should be mentioned that most users had around the mentioned screen resolutions but there were still quite a lot of visits from desktop users with a screen resolutions as far down as 1024x768 points. In fact 1024x768 points was the tenth most common resolution of the visitors. On the other hand the visitors with this screen resolution were deemed to be consumers as they mostly visited the page (mentioned as *Page A* above)that had been identified as being visited primarily of consumers.

By excluding data where the landing page was *Page A*, which was the primary attraction for consumers during the data period, a somewhat different picture arose. The most common resolution was then 1920x1080 points at 7 % of visits in the data set which excluded visits to *Page A*. The 1920x1080 points resolution was closely followed by 1366x768, 1440x900 and 1280x800 points each at around 5 % of the visits. With a wide array of different resolutions each at a low percentage, most of which were also desktop or laptop sized. As a consequence, it can be concluded that the developer site is primarily visited using either a screen of medium to large size.

4.1.3.3 Location, date and time

Higher bounce rate and lower engagement could be seen for users that were not from English speaking countries. Especially from Spanish speaking countries. This could be due to a language barrier, but not guaranteed, which in turn is primarily a content issue given the nature of the site. To support those who are not good enough at English to absorb the information a translation of said content would be required. As this study does not deal with the actual content but rather the chrome around it section 1.2 we did not go further into the problem area than to indicate that it could very well be a real

problem and that anyone interested in also improving their content should have a look at these metrics.

On the other hand there were some major differences in traffic volume depending on day of the week while the hour of the day seems mostly constant except for the night and morning where traffic decreased to about a third. This type of information is more interesting for this study than the location and language of the visitors as it can be an indication of the context of use. The largest chunks of traffic were seen during weekdays, at roughly the same amount of visits per day, while the weekend saw about 20 % less visitors a day yet still a rather significant amount of visits. Therefore it seems likely that the site is being used quite frequently by both people in an office or work environment as well as hobbyists.

4.1.3.4 The home page

When looking at the home page, which could possibly be one of the main entry ports to the site, a higher bounce rate could unlike for the rest of the site be a negative aspect as the home page's main goal is often to funnel the visitor to the information or service she is looking for (Nielsen and Loranger, 2006). The opposite could of course be true in the case where the home page contains the entire site's content. A completely valid and possible scenario all depending on the design of the site. In the case of the Spotify developer site this was not the case, the page instead lead to specific development resources for the different products. Therefor when analysing the behavior of visitors entering via the home page it was determined that a bounce was to be perceived as a negative event. Signaling that the visitor did not find what she was looking for or a way to get to that content.

One of the major problems with the design of the home page at the time of the study was placement of links and call to actions. The three primary navigation elements, besides the main navigation bar at the top, which were styled as clickable elements were visible to less than 48 % of visitors to the page without scrolling. This as the average resolution height was 801 points and the button like links were laid out 824 points from the top. Given the finding by Nielsen and Loranger that only 23 % of visitors scroll the first time they visit the page these such important buttons need to be placed higher. As to optimize the amount of interaction with the website and thusly computing device to get to where she wants. If one would want to cover about 90 % of visitors a good target would be to support a minimum resolution of 620 points high and 1004 points wide. For 95 % of visitors the minimum visible area shrinks to 569 points high and 986 points wide. This data was acquired using Google Analytics' *in-page analytics* tool which shows, using a colored overlay, the area which can be seen without scrolling by percentage of visitors (see figure 4.14 for an example).

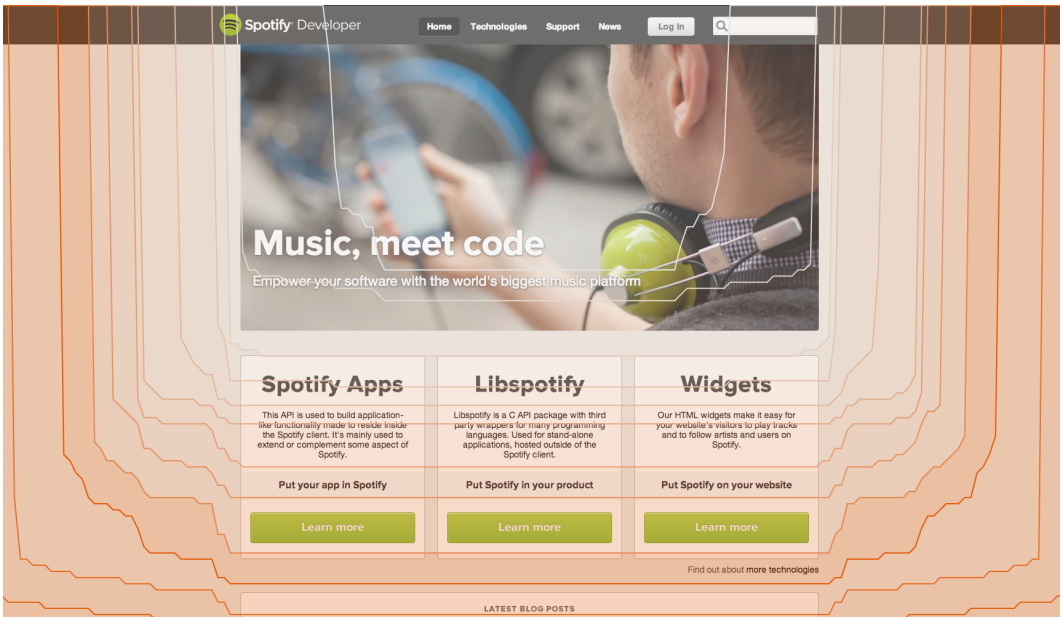


Figure 4.14: An image showing the portions of the home page which are visible without scrolling to multiple percentages of visitors. *Note that the box titled "Widgets" in the image previously had the title "Play Button" and a different description but did look the same. The data used to draw the visible areas are the same as for the rest of the study, i.e. from 2013-03-01 through 2013-08-31. The image was generated using Google Analytics.*

4.2 User results

In this section we present the results from user research and persona development. Thereafter we present an aim to identify *what* a developer page actually is for our target group.

4.2.1 Research results

Interviews were carried out with stakeholders from both Spotify as well as external developers. As previously mentioned in section 3.3 we identified three relevant target groups; the professional developer, the novice developer and the tester. This led to interviewing quality assistance testers at Spotify, intermediate developers at a Spotify hackathon comeptition and external professional developers.

Interviews with testers at Spotify were carried out in a semi-structured manner during the period of a week. Based on a set of prepared topics we formulated start-off questions but then let them formulate their own stories. We discovered that not all were very technically skilled. About two of them were technically advanced and three were average.

Thus we generalize the tester as a slightly above average technically skilled person. When it comes to the Spotify platform, of course they have good knowledge. Since they almost daily visit the platform and on top of that are able to affect the content, they are well informed. They know it so well that even though they can point out bad features to it, they either have workarounds or other additional resources where they can seek for information. This puts them in entirely different positions that our remaining target users. Nonetheless are their needs important to us. Personality-wise they are all very service minded, well-structured and social. Important for them is possesing good feedback skills as it facilitates their workload. Therefore they are good at adjusting their communcation level to suit the receiver and formulating feedback in a constructive manner. As testers they work very goal-oriented and value high quality. Since their work tends to be slightly repetitive they appreciate consistency, rules and defined processes. However, when faced with new problems they are generally fast at adapting and learning. Testers at Spotify are used to multiple parallell processes and easily switching contexts. It all comes down to efficiency and quality. Proactively improving processes, prioritizing, measuring progress and completing a goal while not deteriorating the quality of their output nor the time effort.

During two full days we met approximately 100 intermediate developers at the Way Out West Hackathon which Spotify co-hosted. We carried out over-the-shoulder observations and questions while they were coding and occasionally interrupted them for short informal interviews. Participants of the hackathon were from a wide range of technical background, from the very inexperienced novice programmer to the recently become computer science graduate. The average participant was a computer science student half-way through her studies with developing skills in several languages on a general level. She was not professional within any certain language but has a prefered one. When it comes to experienced withing the Spotify platform most of the users had not interacted with it before, and those who had had little continuous experience. Since they do not know their way around the platform they spend a lot of time searching in different ways instead of reading through all the site. Although they are very curious in learning new things, they are slightly impatient in finding answers. They are used to working with tight deadlines and collaborating in teams. Despite this it was clear that isolating themselves in order to ahieve full focus was important too. Although off-site competition required them to bring transportable computers, it was our impression that they work with limited technical resources.

We met two professional developers not employed by Spotify for semi-structured interviews that lasted, en hour each. Some of our conclusions were also based from interviews with two highly technical testers we met from Spotify. Or professional developers were not inly expeirenced in programming languages but also from different work methods, environements, work areas and people. They showed both high knowledge and interest in technology, both software and hardware. Not only were they interested, but they also caught up on new technology fast. Both were social and were used to working in both large and small teams. Although searching online for information was standard

they really appreciate being able to ask colleagues for help. During their spare time they excersiced a lot and nurture their passion for music. Quality is highly valued and therefore they like to firmly root their decisions before commiting to a certain technology. Being accurate, well-informed and have a good understanding of how things work on both high and low level is important. It is thus likely that they will read up on information and complete tutorials in order to gain full comprehension of the tool. At the same time they need detailed and advanced information. Should a tool not fit their needs, they will read up on another. Should they not find a suitable one then it is not unlikely that they build a solution to the problem themselves. When working they prefer to get into a flow and not be disturbed since they are working on commission wih resposibility and thus have lite spare time for interruptions or unrelated occupations.

4.2.2 Personas

From studying developer psychology and through interview conclusions we identified some values that were important to our users. These were plotted on to a behavioral parameter scale as described in personas (3.3).

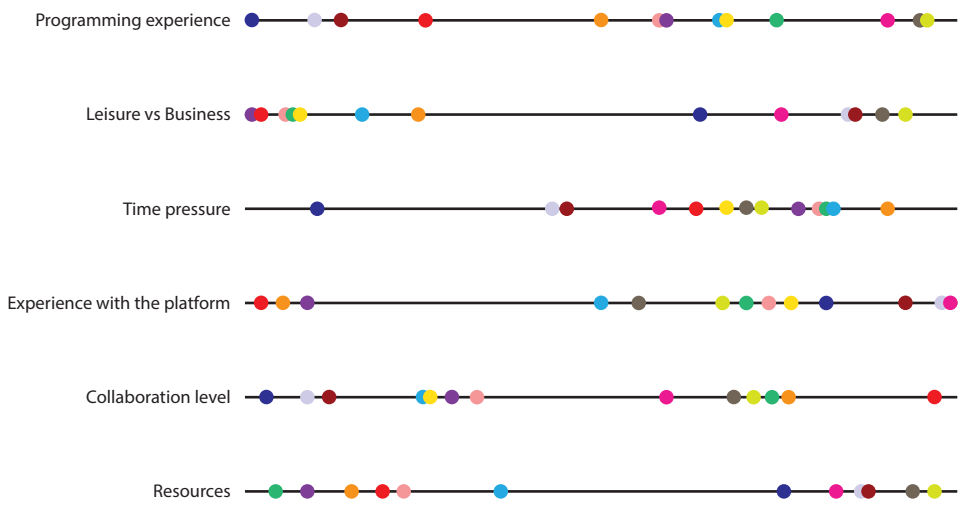


Figure 4.15: Behavioral parameter scale with data plots

Identifying patterns amongst the values led to the following groupings:

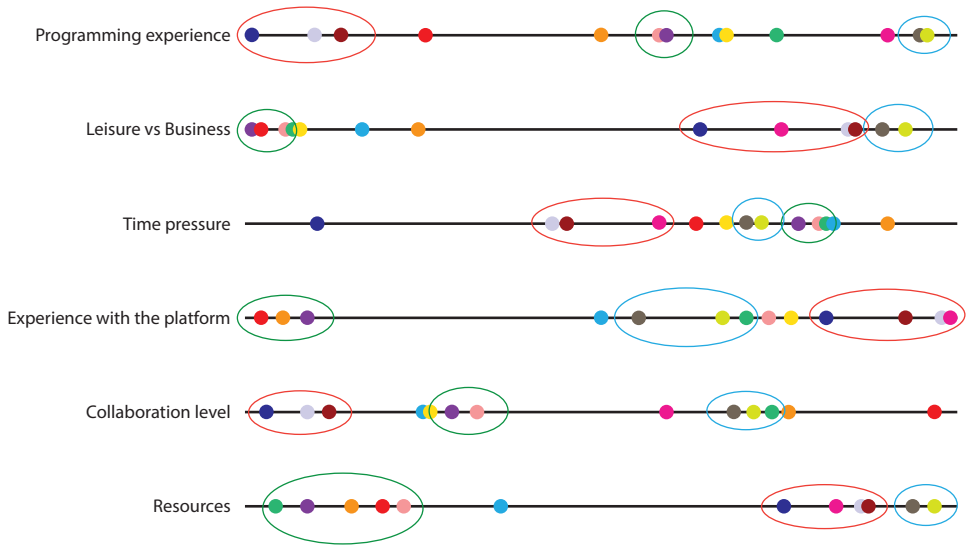


Figure 4.16: Behavioral parameter scale with associated data plots clustered together

We approached the developed user scenarios (3.3) with the user data in mind and defined the different solutions that would be made. From these ways to approach scenarios we defined three personas who would represent our target users. The persona Tara (see figure figure 4.17) is a tester who easily empathizes with others and is a good communicator and service-minded. David (see figure figure 4.18) is a professional developer who values understanding technology or tools thoroughly in order to assure quality delivery. Lastly we have Simon (see figure figure 4.19) who is a computer science student who is passionate about new technology and spends spare time indulging into programming projects.

The results from creating personas aided the design of our prototype. It contributed with a set of requirements, checking design decisions against them and help in prioritizing features. From the personas we got requirements such as easy referencing, good explanations, providing good documentation and code examples, explaining the structure of the site, showcasing inspirational projects, etc.

Figure 4.17: Persona of Spotify QA tester. For higher resolution see appendix C

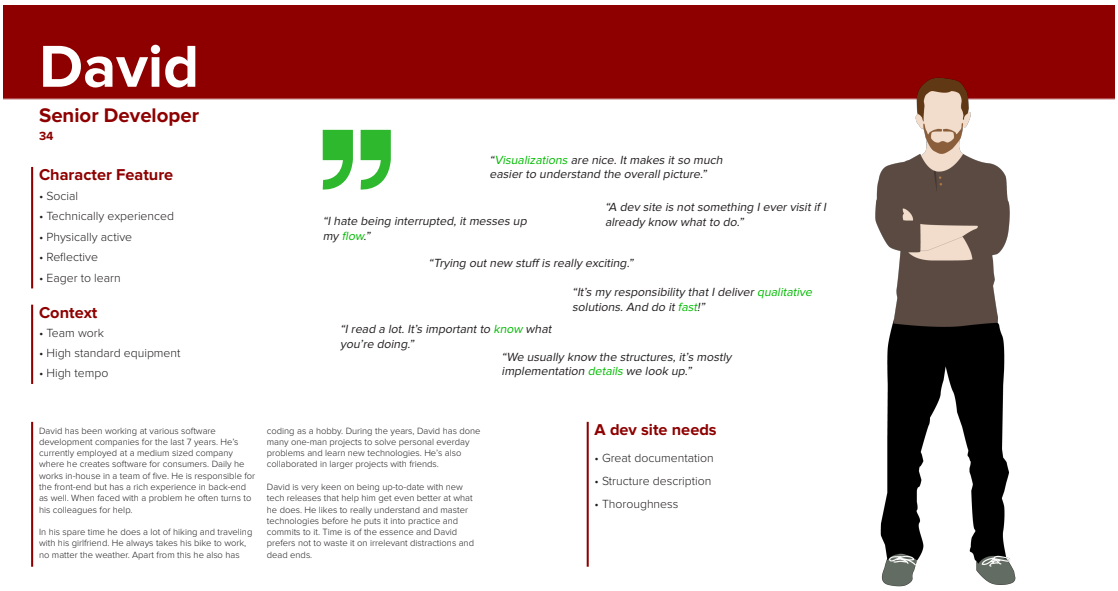


Figure 4.18: Persona of professional developer. For higher resolution see appendix C

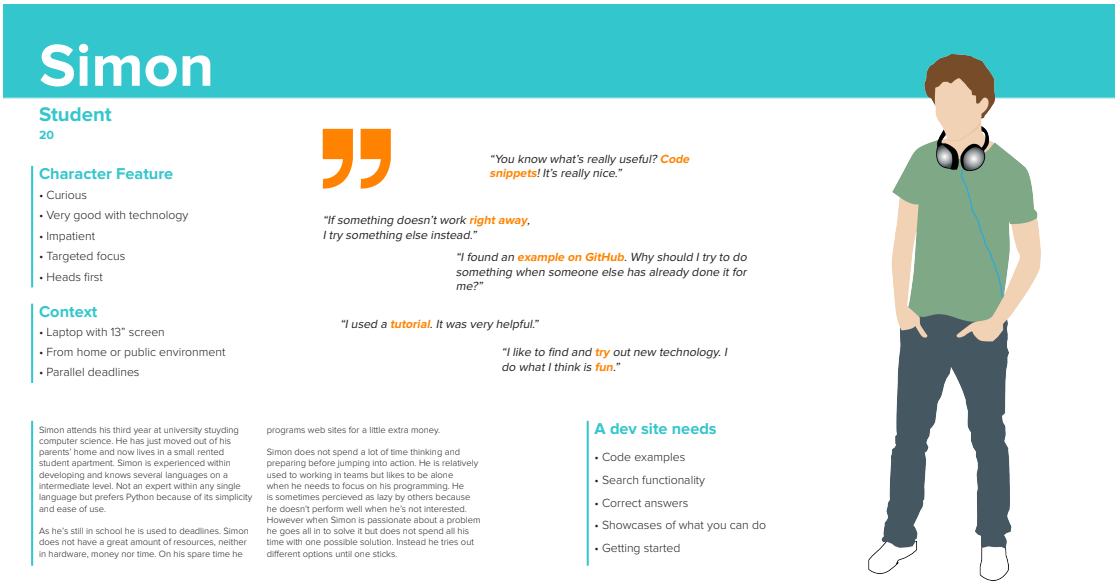


Figure 4.19: Persona of intermediate developer. For higher resolution see appendix C

4.2.3 7 principles

From studying developer psychology and through interview conclusions we identified some values that were important to our users. Based on all previous research and field studies we carried out a brainstorm session where the aim was to define what the purpose of a developer site should provide. The result is divided into 7 different types of principles. What is applicable to a developer site of course depends on the type and purpose of that particular site and the ability to maintain it.

General information

A developer site should provide some kind of high level information about what services are provided and why it is worth investing in them. This should be adjusted to fit the aimed for target group. Thus providing general information on how to get started or a value proposition for investors helps explain the overall purpose of the site. In order to invoke a feeling of trust and assurance it can be good to show that the site is up-to-date with either API health, news section, last updated information or something of that kind.

Problem solving

Sooner or later the developer will run into problems and search for solutions. Providing both general and very detailed help has been shown vital when designing for our personas. A high level FAQ could solve many re-occurring issues without forcing the user to filter through a lot of information. To further avoid the need for filtering, a good natural

language search is as above mentioned beneficial. At times it can be hard to formulate the actual problem. A developer site could help the user express herself and if still no answer is found, then provide the tools to move forward. Connecting the problem location and answer source could be done by integrating support into the crash reporter. We want to invoke a feeling of peace of mind and closure.

Inspiration
When "why should I do?" has been answered the new question users might ask is "what can I do?" Showing off deferent levels or types of inspiration will meet the attention of different users. Identify the most important groups and consider inspirational cases to display. Success stories is a further incentive as to why a user should invest her time. Making sure that a developer site clearly states its different components and functionalities is like showing a set of different ingredients to a chef. The mind immediately starts figuring what possible compositions can be made. Inspiration is not only limited to the final product or outcome but can also enlighten the user about the process and how to approach this. Explanatory graphics and perhaps also interactive elements is appreciated by developers and makes it easier to understand relations between components.

Resources
A developer site should be like a programmer's bible to that technology. It should contain everything useful. If relevant information is stored elsewhere than the site then the site should at least refer to this place. However, the aim should be to keep all resources gathered in one place. Resource collection should be to the point, the user wants to understand what she is viewing and where to find the tools she needs. Items like starter kits, API downloads, implementation examples, editor settings description, tutorials, etc. are good to include in the site. There are two ways of grouping resources, either dependent on their type or on their category. For example grouping hot dogs and hamburgers, either they are grouped bread versus meat or according to what dish they are. From our experience grouping by category separately is to prefer. Users will seldom move from one category to the other but rather search for information within the same area. This means for example grouping libSpotify API together with other libSpotify resources instead of grouping all sorts of API types together. An explanatory mindmap over the structuring helps the user grasp how the page is organized.

Forum
A developer page is also a natural place for a developer to look for expert help. Discussing ideas or problems with other experienced or employees can serve as a form of mentoring. This requires a lot of manpower to maintain but can also contribute to a large collaborative collection of knowledge. It can not only provide good information between developers but also back to the organization. For example if many re-occurring topics on a fix appear, then the users have identified a problem worth fixing and perhaps also a solution.

Meeting place
Meeting like-minded others can provide for a sense of belonging to a community. Important for developers is learning and being inspired by others. If a developer site feels inviting

and makes the user feel like one in the gang then the likelihood that she will get attached to and contribute to that community is higher. An online meeting place might lead to physical meetups which furthers strengthens the feeling of cohesiveness.

Sharing
Although quite similar to the previous two principles we still point out that a developer page can serve as a hub of sharing, receiving feedback or just publishing work. Providing a form of "app console" gives the user the control and overview of shared content.

4.3 Prototype of Spotify's developer site

Given the theory and research performed a prototype of how Spotify's developer site could be designed was created. In this section and its subsections the decisions based on the requirements found as well as contents of Spotify's developer site which lead to the prototype will be presented, alongside the mockups of the prototype. The major areas as well as details which were identified as particularly important will also be gone through in more detail. All mockups can be view in full resolution in appendix D.

It should be noted that the content of the original site was not changed in any major way. Consequently content which was already available from the original site was used and only content needed for completely new parts were added.

4.3.1 General design decisions

Based on the discussion in the theory section on branding (2.4.1) and brand alignment (2.4.2) we decided to quite strictly follow some of Spotify's brand guidelines. With the intent of evoking a feeling of trust as well as making it easier for the visitor to identify what site she is on and who is behind it. The elements which were followed most closely was; the use and placement of the site identifier (how it should look), the typeface used as well as the overall color scheme.

The site identifier style Spotify use is to place the subsite's name (in this case "Developer") to the right of the vertical Spotify logo as can be seen in figure 4.20. We chose to do this such that the visitor would be able to directly both identify the organization behind the site as well as the specific subsite and its purpose. More so, these and all of the elements on the site used only the brand colors associated with Spotify. Except an additional grey and the colors of a few minor items, the latter will be discussed further in the section on search (4.3.4. The reason for the choice of color scheme was once again to boost the coherency with the main Spotify site and offerings, and create a feeling of authority. With the intent of insuring the visitor that the information on the site was

Figure 4.20: Showing the site identifier used.

the one true set of information. The typeface used, namely the sans-serif font *Proxima Nova* by *Mark Simonson*, was selected for the same reasons as the colors matching the organization's brand guidelines. It was applied to all textual elements spanning from headings to body text and minor details. The only exception being code listings for which the typeface was not optimal due to the properties of code and what developer expect based on to habits. For such elements a monospaced font, *Menlo* by *Jim Lyles*, was chosen as it works well with the primary font. The primary font communicates a feeling of openness, helpfulness yet being modern. Trying to convey the message or sense that the site really is there to help the developer and improve on what she is doing.

Some deviations from Spotify's style was however made as not all of the fit the purpose of a developer site and could thus be worse for usability. A matter discussed by Cooper et al. (2007) and exemplified by Facebook's developer site[5] which matches the original brand so well that it might be difficult to understand whether one is on Facebook's main site and using that service or visiting the developer site.

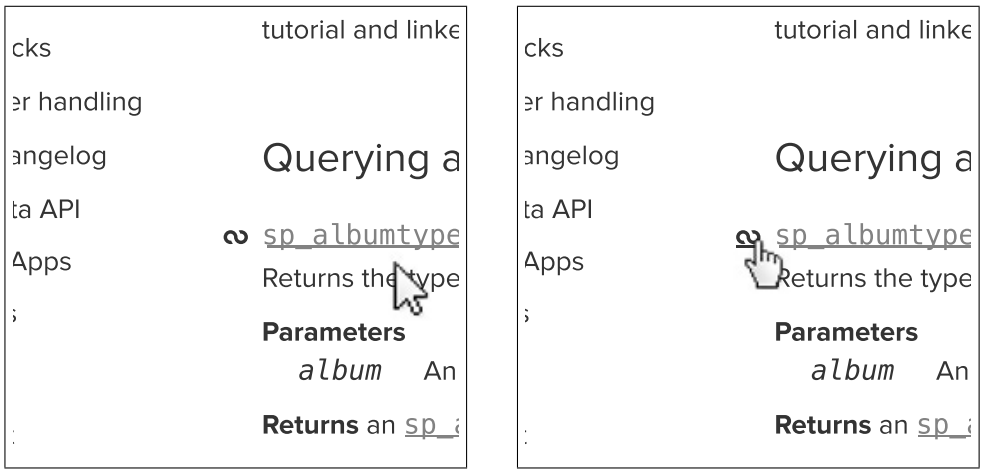
4.3.1.1 Section links

One special case which became apparent in the interviews performed was that a way to reference or link to specific content would be greatly beneficial for several of our target groups. Perhaps especially so for those trying to help others in either official support or via some forum or public questions and answer site (such as *StackOverflow*). To accommodate for this need a *section link* element was placed to the left of each heading. For it to not take up too much visual attention and draw the user away from the content the link element was made to only be visible when the user hovered that section, as can be seen in figure 4.21a. Hovering the actual section link would also underline the element to show that it was a link element and actionable (see figure 4.21b).

4.3.1.2 Related content

Another important form of linking employed was that of linking related content. It became evident during the competitive review and subsequent interviews that this would help users of the site find out more about any given topic. An example of this can be seen in figure 4.22 where the module (*Album* of the libSpotify API) itself has a set of related pages. The individual members (that is functions, types, constants and so forth) could in turn also have related content such as tutorials, other members or any other type of content on the site. The boxes showing related content had been placed in the margin of the page to signal that it might not require the highest amount of attention while still being attached to the item left of the box. To the left of each related item link the same icon which was used in the search results and suggestions (see section 4.3.4)

[5] `https://developers.facebook.com`: At least at the time of writing this report, 2014-03-24



(a) Hovering the section shows the section link.

(b) Hovering the section link underlines it.

Figure 4.21: The figure shows the two hover states of the section link for a *libSpotify* function with the return type `sp_albumtype`.

was used, although it was not colorized on in the related content box as that would have taken occupied too much visual space.

4.3.2 Site layout

The general layout of the site was decided upon based on the types of content, information architecture (discussed in section 4.3.3) and brand alignment. The most important factor for the developer site was the content and being able to find what one was looking for. As such the content was put at the very center with a floating header at the top and a footer at the end of it (non-floating). The main menu used for navigation was in turn placed to the left of the content. This was the location where the largest fraction of users expected it to be, given a left to right language like English which the site was written in. See figure 4.23 for an overview of the layout. In this section the header and footer will be discussed in greater detail while the menu and navigation structure will be discussed in its own section (4.3.3). The reason why the content was placed in the center of the viewport, although inset by the main menu, is because it was found to be the location where user's look for it (Nielsen, 2006).

One deviation from Spotify's branding guidelines of the prototype was the looks of the site header element. Spotify used a dark background with light text on it however the content on the developer site needed much more visual space (not specifically more physical space) than most of other sites

Figure 4.23: Showing the overall site layout, the orange element at the top signifies the header, the green element to the left the nav-

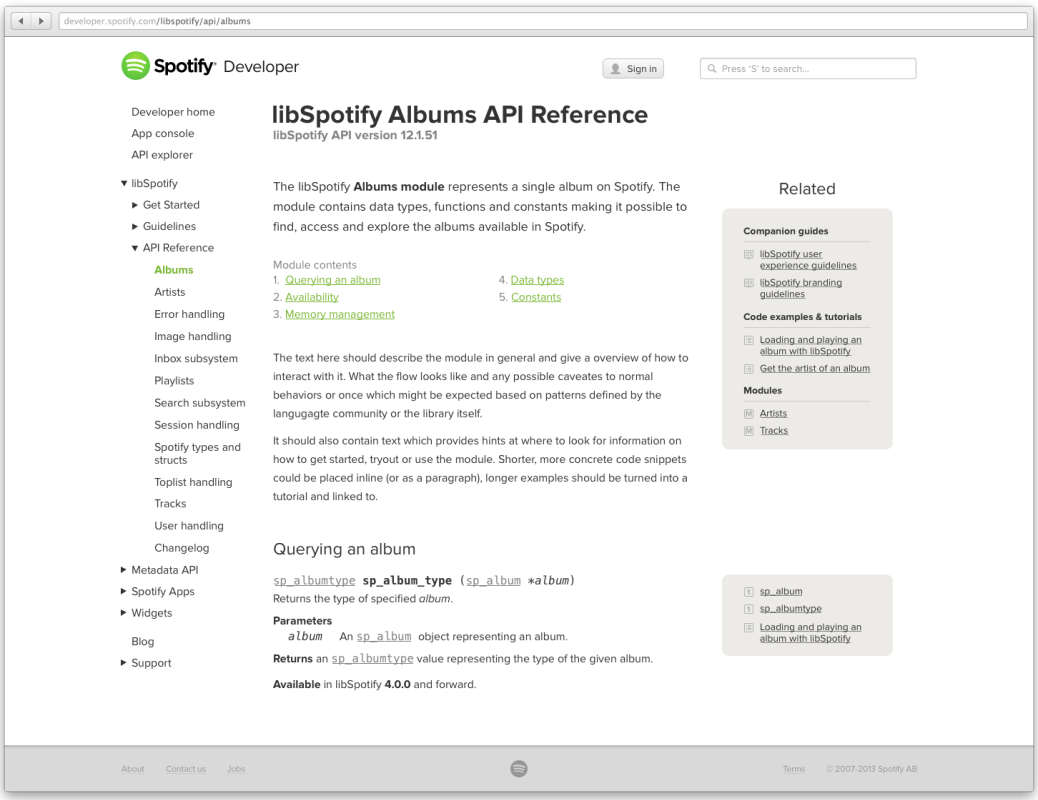


Figure 4.22: Showing the API reference for the libSpotify Album module which has a set of different related content. These related pages and content is shown to the right of the normal content.

under the Spotify umbrella. A decision to deviate and use a white background instead with dark elements (as can be seen in figure 4.24) was made, this allowed the header to more easily blend into the surroundings. Although the color scheme was altered the layout was kept mostly intact as it was also supported by our findings in user research (section 4.2.1) and in literature (section 2.6). The logo was kept on the left hand side of the site, as the language was left to right, meaning that this was the location where most users expected it. The search field was in turn also placed where most expected it given a left to right language. Namely to the right, aligned with the right hand side content edge. This field, as well as the search results page, will be discussed further in the section on search (4.3.4). To the left of the search field the account button was placed as this is the location used by

Spotify. Such a placement allowed for people accustomed to how Spotify's other sites worked also feel at home on the developer site.

Figure 4.24: Showing the site header with the site identifier on the left hand side, the power mode button at the far right, a search field to the left of the power mode button and then an account (switches between "sign in" and "sign out" depending on the visitor's signed in state) button to the left of the search field. This is aligned with the layout of Spotify's main site and subsites for the most part.

At the very far right, just after the search field and outside the right hand side content edge, a power mode toggle button was placed. Like the name suggests the feature was intended for power users of the developer site which was also why it was placed as far out as it was. That is for those users whom are already accustomed to the material available and crave an even lower lookup time and speed of the site with shortcuts but do not fear investing a great deal of time. Or as Cooper et al. (2007) would have put it the "expert" users. Its placement made sure that it was still findable yet not in the way or taking up too much space for the vast majority of people who would never use it.

At the opposite end of each page an element aptly named *footer* (see figure 4.25 for a visual representation) was placed to contain some minor navigational elements and also mark the end of each page. On the left hand side of the footer, navigational elements which concern details about the company and ways of getting in touch were placed as these types of items were expected by users (section 2.6.1). After the two more common links, and also more or less mandatory according to literature, "About" and "Contact us" a third link was placed. Namely "Jobs" which would take the user to Spotify's jobs section. Spotify was a product development company itself which was always looking for new talent. As the visitor was already visiting the developer site of said company we argued that it was reasonable that they might also be interested in working there. The opportunity of finding talented and interested candidates was too great to miss thus the link felt like a must.

Figure 4.25: Showing the site footer with the Spotify logo in the center, clicking it takes the user to the main Spotify site (i.e. `spotify.com`). To the left of the logo three links have been placed; "About", "Contact us" and "Jobs". To the right of the logo the legalese have in turn been placed with a link to terms and conditions as well as a label displaying the copyright information.

In the middle of the footer Spotify's logo was placed and then also linked up with the main Spotify site as a connector between the two sites. Going between subsites and the main site can be somewhat taxing if the user has to directly manipulate the URL (*uniform resource locator*) in the browser's address field instead of clicking a link. It also enhances the connection between the developer site and the organization behind it. This feature was found during competitive review of *GitHub*'s developer site which employs an identical approach.

Lastly the footer contained some legalese at the far right which was also found to be a required piece of information, both from the user's point of view but perhaps primarily from the organization. This consisted of a link to all the terms and conditions for using the services and tools provided as well as a copyright notice.

4.3.3 Navigation

As one of the most important goals set out for the site was the speed (how quick the visitor could find and attain the information, resource or answer sought) we considered several forms of navigational structures. Ranging from have a navigation with drop down menus at the top putting the main menu at the center of the page. The latter at least on the home page as its goal was to funnel the user to the correct location, more on that specific page can be read in section 4.3.5. Given the services, tools and products (also referenced to as technologies) offered to developers by Spotify, an architecture over the site was created and can be seen in figure 4.26. The site was categorized into a few top-level items which were either a single page or a category in its own, these were;

- Developer home (the home page)
- App console
- API explorer
- Blog
- Support
- *libSpotify* (technology)
- *Metadata API* (technology and service).
- *Spotify Apps* (technology).
- *Widgets* (group of technologies).

The technology categories *Spotify Apps* and *libSpotify* had very similar content and their structures were consequently made to be identical for the least amount of friction if a visitor visited both. They were both organized into three different sub-sections;

Guidelines: Which contained any guide or guideline documents associated with the parent section. For example a *user experience guideline* or *integration guideline*.

Get started. Which contained tutorials for getting started with the technology. As well as tutorials for advancing the visitor's skills and learn new ways of using the technology.

API reference. Which contained the information about each module in the technology's API and its members, types and constants.

The *Widgets* and *Metadata API* sections on the other hand each had unique and distinct content. Consequently the architecture for each of those sections also became unique. Fortunately only two sub-pages were needed for each of them and a needlessly complex IA could be avoided. Especially for the Metadata API where the sub-pages were named *Get started* and *API reference* just like for the Spotify Apps and libSpotify sections. Thereby providing some similarity and coherence.

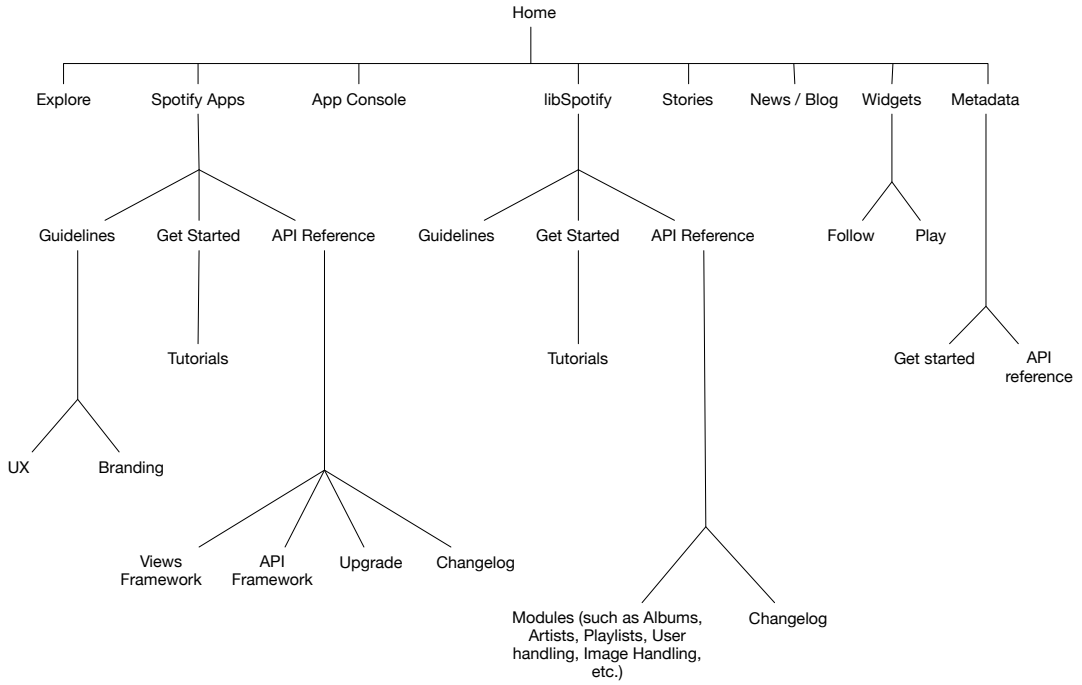


Figure 4.26: The information architecture created for the prototype site, based on the offerings to developers by Spotify.

Given this information architecture (*IA*) it was decided that a navigation to the left of the content would take up the least amount of space. As the site analytics for Spotify's developer site indicated that most visitors had a screen which was wider than tall. Additionally it was decided that the best fit would also be for the menu to float, that is stay in place as the user scrolled. Partly as it would allow the visitor to explore the site from wherever she might be on the site, regardless of the current page or scrollpoint. Partly as it could then also serve as a sort of breadcrumb showing said visitor where

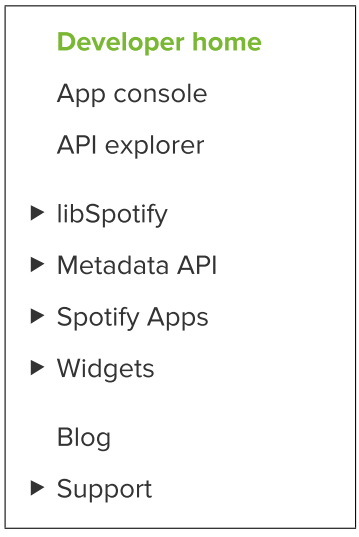
she currently is. This decision and design was made possible due to the relatively small set of top-level pages needed given a categorization by technology, that is *libSpotify*, the *Metadata API*, *Spotify Apps* and *Widgets*.

When a specific category was viewed, or a page belonging to it, the category was expanded showing all of its sub-categories and pages. Each category item could also be expanded by clicking the triangle to the left of the text. See figure 4.27 for the looks of the main menu both in its completely collapsed state (figure 4.27a) and also when several categories have been opened up (figure 4.27b). Clicking any item in the menu, whether a category or a page, would take the visitor to a page so that there would be no confusion of what an item in the menu would do. Sub-categories and pages within another category were in turn indented to show that they belonged to the category above it. As the site was re-organized and re-architectured the indentation of menu items did not start to encroach on the content area due to an ever increasing number of sub-categories. If more sub-categories were to be added in the future the menu design might have to be reconsidered. Although a better strategy would instead be to look over the IA and aim for a broad information architecture rather than a deep. Because a broad and shallow IA makes it easier for people to build a mental model around and understand how to get to another page or where she is.

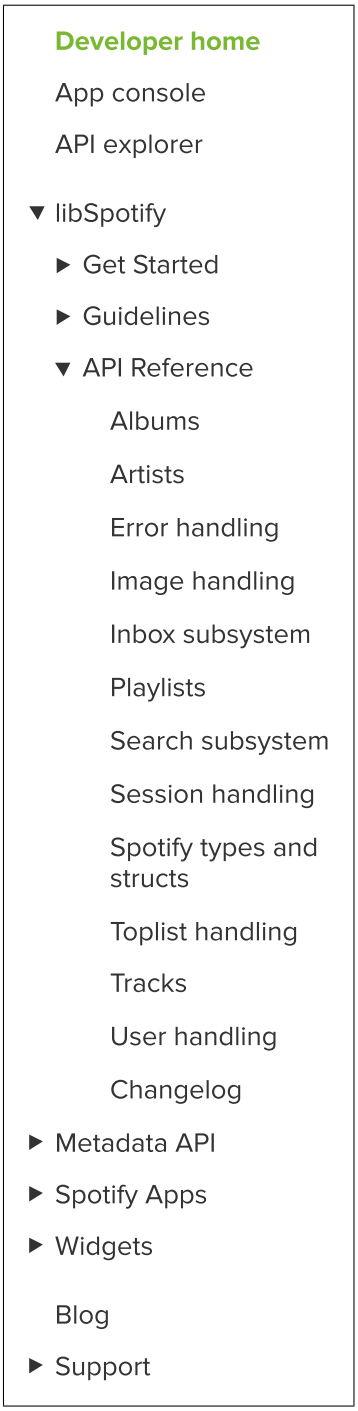
The major problem with this design had to do with the height of the menu relatively to the height of the browser window. As the number of items in the menu increased (by the user expanding or delving deeper into categories) or the browser window shrunk the space available would eventually run out. By looking at the web analytics (section 4.1.3) it was decided that it would not be experienced by enough people to outweigh the benefits of an always available navigational structure (see section 2.6.1). To make it more apparent which page was selected two visual cues were employed, changing the text color to Spotify's primary brand color as well as setting it in a bold version of the font used.

4.3.4 Search

As mentioned in the theory section on search (2.6.2) and then also noted in the interviews performed the search functionality is another, important, form of navigation especially for finding specific resources. Spotify's developer site mostly consisted of *API* (application programming interface) documentation as well as some resources, online tools, long form text and some text with a special structure (for example a changelog). As such we felt that the search functionality should reflect this and also provide suggestions and results which took advantage of knowing what each item was as well as how it was structured. Instead of trying to parse all content as if it was body copy like most other sites and search engines seen tried. Thus providing not only better search results by understanding how to search all data but also understanding how to present it in a way such that the user might not even have to click through the page of the match. One example of a site found that did provide something like this was Google's *Android developer* site. Where



(a) Showing the site navigation in its default, collapsed form, as seen on for example the home page.



(b) Showing the site navigation in its expanded form.

Figure 4.27: The figures show the main menu used for navigating the site.

visitors can search not only for textual matches but also for classes and methods in the libraries.

Another goal was for the user to have to jump through as few pages as possible and not have to scroll to get to the desired content. As such all search results were defined to take the user to the location, on the containing page, of the match. Furthermore a search field was placed in the header, as described in section 4.3.2, of each page (see figure 4.29). As the user searched using this field suggestions were automatically shown below it in a popover (see figure 4.28). The items were grouped by technology so that the user could focus on only the relevant results. This could further be refined by for example writing "`libspotify:`" at the beginning of the query which would only show matches related to *libSpotify*. If items were found in multiple sections the number of items per section was reduced down to a minimum of three suggestions per section in order to not create a truly massive list yet show a meaningful amount. On the other hand if results were found in only one section up to eight different suggestions from that section were shown. Lastly an option to show all results was placed at the bottom, providing a way for those whom it might not be clear that pressing enter while in the search field would perform the same action.

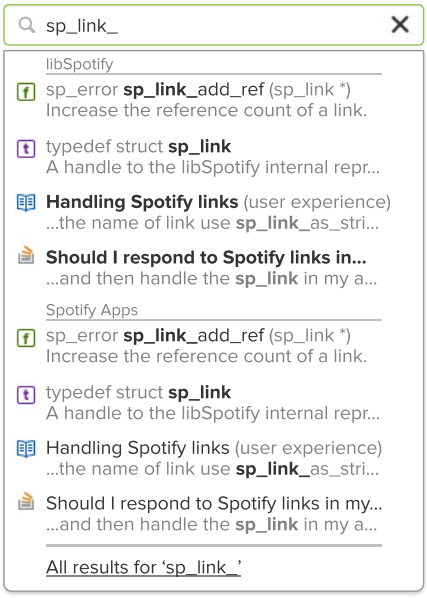


Figure 4.28: Showing the global site search with a query "sp_link_" inputted and suggestions then presented for this query.

(a) Normal, unselected, state. **(b)** Selected state.

Figure 4.29: The figures show the the site's global search field..

The suggestions were then navigable, taking the user to that specific content, by using the up and down arrow keys on the keyboard to select a specific item and then pressing enter to go to it. Alternatively by clicking a specific item directly using the pointing device. Additionally each item was given a specific icon to the left of the title. The intention with this icon was to provide intermediary and expert users with a quick way of determining what type each result was. All API related matches were assigned a square box with an unique color and letter inside. A *function* was color coded green and given a lower case "f" as its character while a *type* was color coded purple and given a lower case "t" as it character, and so on. Guidelines and textual matches where in turn given an icon depicting an open book overlaid with a blue color while a tutorial was given an icon depicting a set of steps and some text. If a match lead to an external site that site's logo was instead used. Most of these icons can be seen in figure 4.28 as well as in figure 4.30.

The actual search term was then highlighted in each search suggestion by bolding the textual match. Each item also had a short description which tried to show the context for the match or some information which was relevant. Different types of results were also presented differently; with guides and tutorials having the heading nearest the match showed as well as which document it belonged to, API references showed the actual definition such that it could be obtained quickly from anywhere on the site without visiting a specific page.

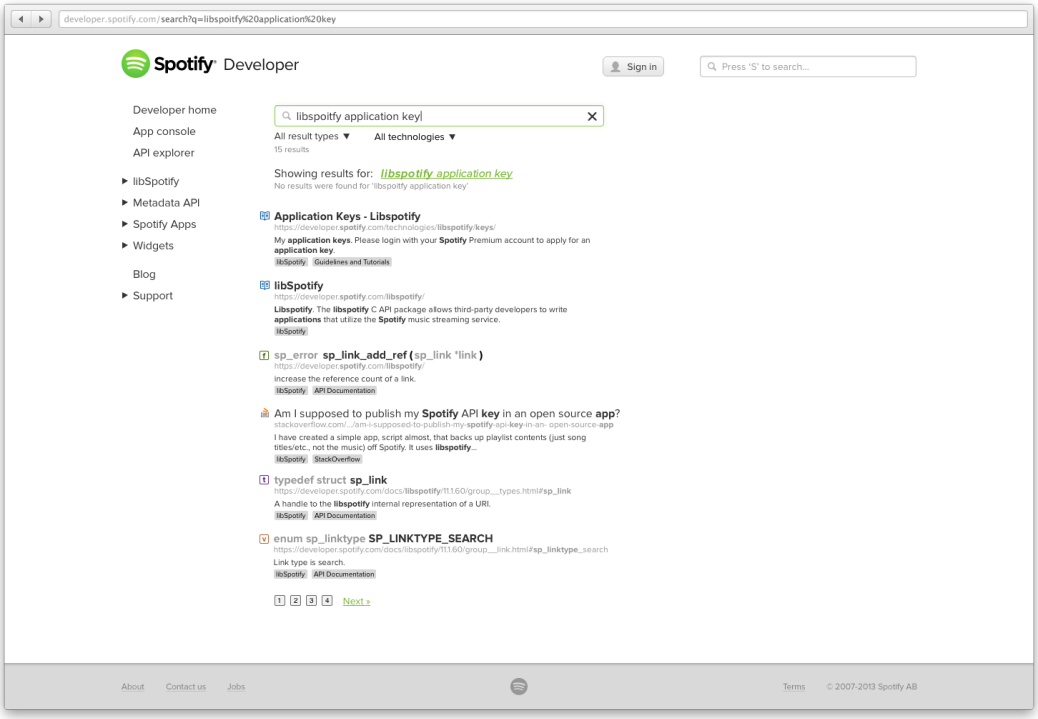


Figure 4.30: The search results page.

The content type hinting and specialization was also used on the dedicated search and search results page (see figure 4.30 for a visual representation). That page also allowed the visitor to refine the search using two different drop down menus to select the type (i.e. all, API member, tutorial, and so on...) of content sought as well as in which section (*libSpotify*, *Metadata API*, *Spotify Apps* or *Widgets*). Additionally the search result page was not limited to any specific amount of results and would instead paginate to accommodate for any set of matches. The search engine would also try to find matches even if the input was misspelled or some form of sensible (for example removing words) manipulation of the search query would be needed to yield results. As space was limited in the auto-suggestion popover it would only provide this data without notifying the user. The search results page on the other hand presents both the number of results found and also whether the query was interpreted or modified in any way to find the matches.

Even though the search engine concept was designed around always trying to produce a relevant set of results there will come a time when this is not possible. Therefore the *empty search result* version of the page would not only show a link to get back to the home page but also try to identify the section in which the user was looking for information. Then provide a highlighted link to that section as well as its high level subsections. There was also links to all of the top sections (*libSpotify*, *Metadata API*, *Spotify Apps* or *Widgets*) such that the user would not end up on a dead page with no where to continue.

Another way to search the site which was found during the competitive analysis was to implement the *OpenSearch* standard.[6] Which makes it possible for users of participating web browsers to search the site without even entering it. All that is required is to enter the url of the site in the browser's address field and then trigger the OpenSearch functionality (often via pressing the *tab* key). An example mockup of how this looks can be seen in figure 4.31.

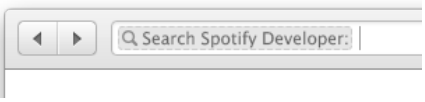


Figure 4.31: Showing how OpenSearch could look in the browser if implemented.

4.3.5 Home page

Several variations of this important page were created in the early sketching phase all with their own flow, layout and unique strengths. Some focused on the difference between Spotify's technologies and used heavy grouping to provide a downwards flow, some focused on the advanced search functionality provided while others on what could be achieved with the technologies. However the sketch which was selected and then subsequently wire-framed and mocked were a relatively traditional version of a home page. The final mockup can be seen in figure 4.32.

In the final version the four technologies offered were put up front and center such that focus would fall on these as the purpose of the home page is to funnel the visitor to what she is looking for. Each of these technologies were represented by an image and its name and then followed by a very brief and to the point description or tagline. The tagline and image was put in as way to quickly inform a new visitor of what each technology provides and which one should be used for different scenarios. As shown in section 2.7 people spend an exceedingly short time on the home page before deciding whether the site serves their need or not. For those who did find the offerings interesting and then at a later stage returned to the site, shortcuts (i.e. links) to the major sections or pages of each technology were placed below the tagline. All still above the so called fold for most visitors. This is also why the introductory title was made relatively compact, disregarding from Spotify's general style of having a large image at the very top. To keep the spotlight on Spotify's technologies high up on the page.

[6] `http://www.opensearch.org/`

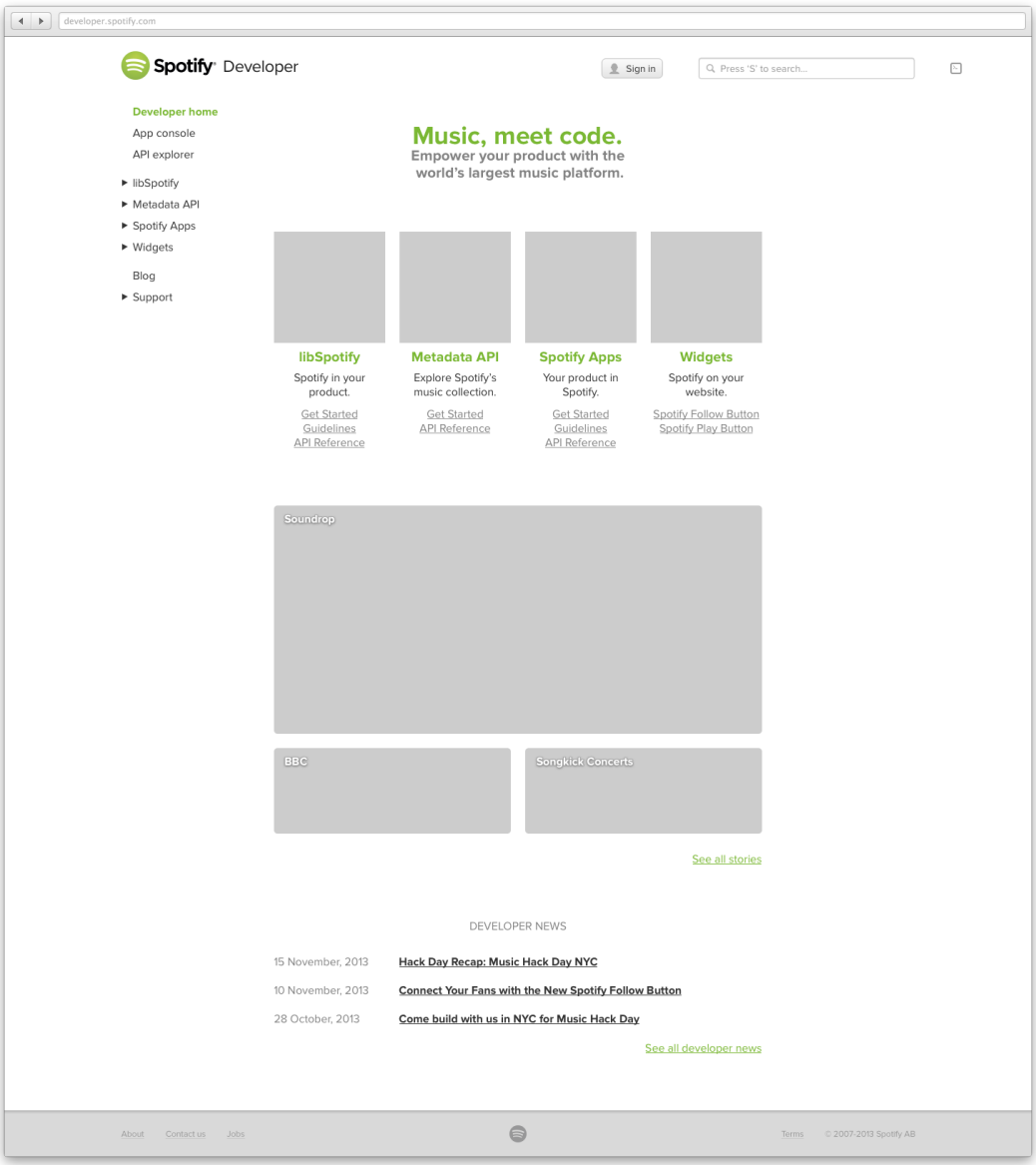


Figure 4.32: Mockup of the home page.

Below the technologies a showcase of what others have used the different technologies for was placed as a way of communicating inspiration as well as trust and community. Initial sketches and ideas had this as a, so called, carousel where the different cases would each be presented up front in a large box and then somehow (most often via a slide animation) be replaced with the next case. Due to the usability issues and reduced efficiency of such an element it was decided to use three static elements instead. First a larger showcase which was then followed by two smaller ones. Such a design do not require the user to

interact with it other than clicking to read more and cause no confusion or irritation when the item she was just reading slides away. A great many edge cases could therefore be avoided altogether. At the end a link to view all stories was placed yet still showcasing a decent amount of stories.

Lastly a short set of the latest news items was placed as a way of showing visitors that the site is alive and the technologies are being worked upon. Thereby trying to instill a feeling of trust that it is safe to use and invest resources into learning and using the technologies and services.

4.3.6 Interactive tools

During user research it was discovered that interactive tools could provide great benefits for several groups of users. Either allowing them to find the answer to a questions faster or by trying out how to use a service. The more common solution to these, which was also seen during analysis of competitors (although most did not directly compete with Spotify), was an online API explorer which will be described in more depth in section 4.3.6.1 below. A *power mode* will also be described briefly in section 4.3.6.2.

4.3.6.1 API explorer

As it was both observed being used, mentioned in interviews and found in the competitive review we decided to add an API explorer to the site. Such a tool allows the user to browse and try out an API directly on the developer site before having to get an access key, download an *SDK* (source development kit) or something similar. Thus minimizing the time needed to evaluate a technology, something sought by the professional developers who might not have a lot of time before having to decide upon using a technology or not. Those found during the competitive review allowed the user to explore and try out the endpoints of services. That is sending a request to a server and then looking at the result. The concept developed during this study built upon that and proposed an extension of it such that it would also be possible to also try out and explore the API of the libraries offered. This would in turn mean that the explorer would need to more or less an online text editor and code interpreter.

4.3.6.2 Power mode

The power mode refers to a special mode, enabled by clicking a button in the top right corner, which changed the site completely. Instead of being a normal site it would act and respond much like a live programming environment on the command line, for example like *erb* which is allowed a programmer to write *Ruby* code directly in her terminal. The target group were only expert users as per the definition by Cooper et al. (2007). The concept was that the user would initiate the mode and then use a pseudo-API to fetch

specific content or resources by just typing. Thus trying to simulate the programmers normal code environment and allow her to never have to leave the keyboard for a pointing device. Nor her thought processes and state of mind while coding. It was hoped that this could improve the efficiency or at least the perceived efficiency. Whether it would or not was not tested as it would have required at least some parts of implementation which was beyond the scope of the study.

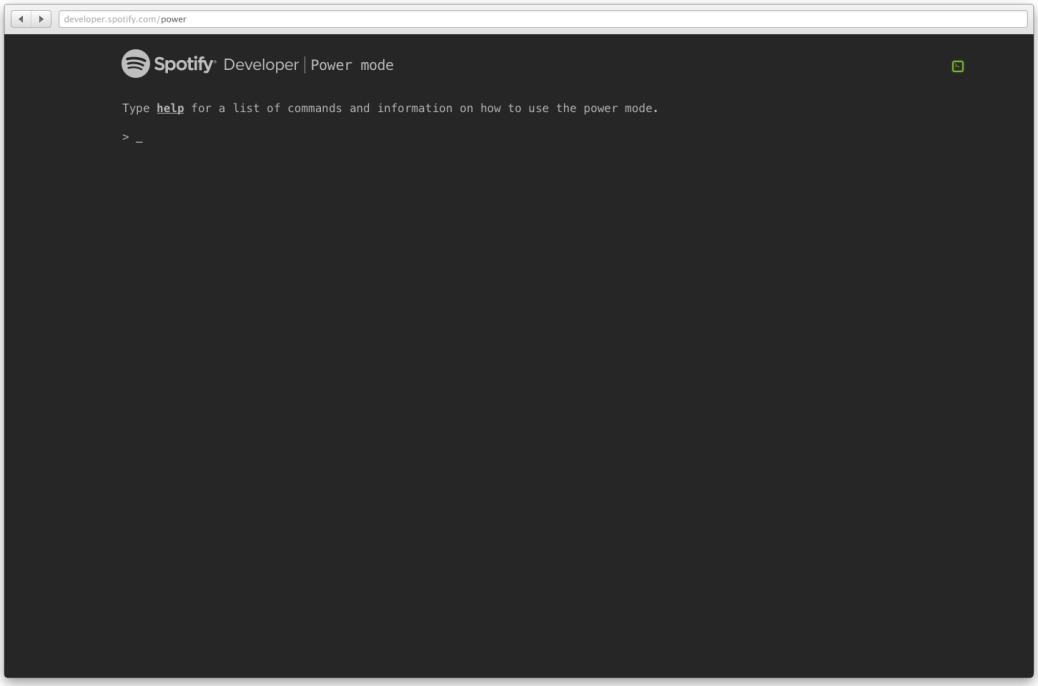


Figure 4.33: Showing the power mode when just entered by the user, via the power mode toggle at in the top right corner. When in the mode, the button glows to show that the mode is active and where to click to exit. Typing `help` while give a list of commands which can be used as well as further information on how to use the tool.

A mockup of the mode can be seen in figure 4.33 which has been modelled after the terminal. The user entered it via clicking a power mode toggle button at the top right corner, to the right of the search field. In the mode all normal site elements such as the navigation, search and footer were removed as these utilitizes could all be accessed by typing. When in need of information on how to use the tool or what commands were available to use could type `help` followed by a press of the enter button. This would then print a list of information. Furthermore the word *help* has been linked such that if the user were using her pointing device it would produce the same output as typing in the command. To exit the mode the user could either click the power mode button again, the site identifier (Spotify Developer) or use the browser's back button (as the mode

makes sure not to break this). Lastly the remaining header elements, site background and textual elements' color scheme was changed to a light-on-dark and the typeface changed to a monospaced (*Menlo*) to mimic the look and feel of the developer's normal environment.

The power mode was created as a way to push the constraints (as discussed in section 2.5) by looking back at the environment and tools developers have used for a long time and still use in their day to day. This mode provides an extra dimension to the site which could be advantageous if implemented but would, as Goodwin (2009) argues, not be a disadvantage if not.

5

Discussion

The following chapter presents discussions about methods we used and the results we present. We attempt to criticize our project from an outside perspective and reason around choices we made.

5.1 247 Questions method

We performed a competitive analysis on a set of developer sites with the use of a method called 247 web usability guidelines by Dr. David Travis (Travis, 2009). Since the test constituted of scoring different aspects of the website it opened up for the risk of personal opinion affecting the outcome. We were two people performing these tests which further obstructs consistency. In order to prevent this we performed the first test together and agreed on what factors would contribute to a certain scoring. However, being consistent was difficult since agreement how to score a certain page was not directly applicable to another. We consulted each other as much as possible to make sure we were on the same page.

Our background might have affected the scoring as well. As interaction designers, we can consider a design feature a bad implementation while a unaware web design tester might not see the issue with it. Thus giving the feature a positive score an relying on the method even though the implementation might be lacking or could be better. To summarize, the 247 web usability guidelines test opens up for subjectability. However we feel that our overall scoring is just and with our appropriate competence in mind should be enough to serve as material for our research.

Furthermore the size of the websites we studied were not consistent. Comparing MSDN to App.net was therefore difficult with this test due to their entirely different situations and amount of content which affects the design approaches.

We modified the test to match our purpose. Giving a test area 0 % in score would disregard that area in the total score. In some cases, such as search, we preferred to

incorporate that area into every test since we saw the importance of it. Lacking a search functionality would therefore lower the scoring.

5.2 User testing

Due to the fact that there was very little prior research on developer psychology and developer resources design we put a lot of focus into exstensive pre-studies and user exploration which was time consuming. We aimed to perform user tests with all our target groups when our prototype was in a test ready phase. Due to the time constraint as a result from the plentiful research studies and the limitation of setting up test environments with the correct users we did not perform these. If we would have had more time and resources we would have prefered to also perform thorough user tests.

5.3 General reflections

Our vision with this thesis was to revolutionize the entire way of approaching interaction between developers and developer sites. We learnt that new is not always better. As a result from extensive research and field studies we saw that some well used methods were actually very good. What we then aimed for was finding out *why* they were good. We also learnt that presenting a new way of interacting would require that the effort for the user to understand and adopt it must be outweighed by the improved user experience. In our case we understood that developer websites are not sites that our users spend a lot of time browsing and was thus not suitable for extensive behavioral redesign. We listened to our users, lowered our expectations and left our egos at the door.

We expected there to be a lot more previous research on the psychology and behavior of developers. The field of human analysis is quite vast and we are prone to try and explain the nature behind certain behavior so naturally we figured that developers would have been studied as they form a great part of today's people. We therefore spent more time on researching this than we had initially planned in order to fully support our design decisions. We hope to gathered relevant current research as well as contributed to this field ourselves.

6

Conclusions

The goal of the thesis was to investigate and formulate how one designs a good user experience for developers when it comes to developer websites. A set of guidelines were presented as well as an example prototype based on those. The project was performed using established methods in accordance to a design based research approach.

Due to lacking existing research on developer psychology we put a lot of effort into understanding their behavior and goals. A general characteristic is that they highly value quality and examples but above all efficency. Time is of the essence to them and they do not want to waste it on trying out approaches that will not solve their problems. They are creative people with the possibility to learn new technology very fast. Not only do they work within development but they also have it as a hobby, a probable reason for why they excel at it.

One of the primary findings was the importance of search. Both by making the content available such that external search engines like *Google* and *Bing* can process it and by providing a good search experience on the site. A large amount of developers start out using a search engine to find the answer to a questions. The site consequently have to be prepared for users starting their journey on any page. It should also be noted that due to this flow of visitors a high bounce rate might not be a negative thing. The event could just mean that the person found what she was looking for and then went back to her project.

To support such behavior and also the context which many developers find themselves in the site must perform well. Meaning that it should be fast to find information and resources. One way to optimize for this and also make it easier for visitors to orient themselves are to design the information architecture to be broad rather than deep. This allows for the user to also move in between sections more easily.

Many of interviewed developers found example code, projects and accompanying tutorials to be a great asset. Making it possible to try out new ways of solving problems and getting things done quickly. These were many times used either for reading and understanding the flow or they were just copied and then used directly in the code. The latter were

mostly be developers new to the platform or library.

When it comes the look and feel of the site it was found that aligning with the organization's normal branding guidelines provided an improved usability. This because the user knew at which site she was on and could thus more easily trust the content to be accurate. Care should however be taken to not let the branding guidelines hamper the usability of the developer site. Certain elements in the guidelines might not be suited for the type of content or flow of developers sites.

7

Future

As this study only focused on the broader strokes of a developer site there are a few areas which could be studied further in the future. To begin with, one area of interest which could require a great deal research is the actual design of the API reference. What should be included, how it should be laid out and linked up. As well how to represent the many different idioms and syntaxes of different programing languages. Perhaps it is not possible to create a single one design instead a list of things to have in mind when design the API reference system targeted at a library or language.

Another aspect which arose during interviews was that of a community or how users could provide feedback and how this should be integrated into the developer site. That is, if it should be integrated at all. Perhaps a better strategy than building one's own community is to utilize an existing one such as *Stack Overflow*. As well as how the organization behind the project should or could handle such feedback, questions and discussion.

Bibliography

Apple Inc (2013), 'Apple Job Creation'. Visited December 11, 2013.
URL: *http://www.apple.com/about/job-creation/*

Barab, S. and Squire, K. (2004), 'Design-based research: Putting a stake in the ground', *The Journal of Learning Sciences* .

Bernard, M., Liao, C. H. and Mills, M. (2001), The effects of font type and size on the legibility and reading time of online text by older adults, *in* 'CHI '01 Extended Abstracts on Human Factors in Computing Systems', CHI EA '01, ACM, New York, NY, USA, pp. 175–176.
URL: *http://doi.acm.org/10.1145/634067.634173*

Bowles, C. and Box, J. (2011), *Undercover User Experience Design*, New Riders.

Brown, T. (2009), 'Designers - think big!'. Visited September 5, 2013.
URL: *http://www.usabilitynet.org/tools/brainstorming.htm*

Brown, T. (2013), *A Pocket Guide to Combining Typefaces*, Five Simple Steps.

Burby, J., Brown, A. and Committee, W. S. (2007), *Web Analytics Definitions*, 4.0 edn, Web Analytics Association, 2300 M Street, Suite 800, Washington DC 20037.

Clarke, S. (2004), 'Measuring API Usability', *Dr. Dobb's Journal Special Windows/.NET Supplement* .

Cober, R. T., Brown, D. J. and Levy, P. E. (2004), 'Form, content, and function: An evaluative methodology for corporate employment web sites', *Human Resource Management* **43**(2-3), 201–218.
URL: *http://dx.doi.org/10.1002/hrm.20015*

Cooper, A., Reimann, R. and Cronin, D. (2007), *About Face 3: The Essentials of Interaction Design*, Wiley Publishing, Inc.

Cutrell, E. and Guan, Z. (2007), An eye-tracking study of information usage in Web search: Variations in target position and contextual snippet length, *in* 'CHI '07:Proceedings of the SIGCHI Conference on Human Factors in Computing Systems'.

Dawn Shaikh, A. and Lenz, K. (2006), 'Where's the Search? Re-examining User Expectations of Web Objects', *Usability News* **8**(1).

Dede, C., Nelson, B., Ketelhut, D. J., Clarke, J. and Bowman, C. (2004), Design-based research strategies for studying situated learning in a multi-user virtual environment,

in 'Proceedings of the 6th international conference on Learning sciences', International Society of the Learning Sciences, pp. 158–165.

Dennis Kardys (n.d.), 'The Ethics of Influence and Manipulation'. Visited October 15, 2013.
URL: *http://www.robotregime.com/index.php/articles/view/influence/*

Djamasbi, S., Siegel, M. and Tullis, T. (2010), 'Generation Y, web design, and eye tracking', *International Journal of Human-Computer Studies* **68**(5), 307 – 323.
URL: *http://www.sciencedirect.com/science/article/pii/S1071581909001918*

Dupuis, E. A., ed. (2003), *Developing Web-Based Instruction : Planning, Designing, Managing and Evaluating for Results*, Neal Schuman Pub. Chapter 8, Usability Tests.

Dyson, M. C. (2004), 'How physical text layout affects reading from screen', *Behaviour & Information Technology* **23**(6), 377–393.

Goodwin, K. (2009), *Designing for the Digital Age: How to Create Human-Centered Products and Services*, Wiley Publishing.

Google Inc. (2013*a*), 'Analytics Help'. Visited October 17, 2013.
URL: *https://support.google.com/analytics/*

Google Inc. (2013*b*), 'Google Analytics Official Website – Web Analytics & Reporting'. Visitor October 29, 2013.
URL: *http://www.google.com/analytics/*

Hasan, L., Morris, A. and Probets, S. (2009), Using google analytics to evaluate the usability of e-commerce sites, *in* M. Kurosu, ed., 'Human Centered Design', Vol. 5619 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 697–706.

Jackson Fox (2008), 'Research-Driven Design'. Visited December 11, 2013.
URL: *http://www.slideshare.net/jacksonfox/research-driven-design*

Kaushik, A. (2007), 'Excellent Analytics Tip #11: Measure Effectiveness Of Your Web Pages'. Visited October 24, 2013.
URL: *http://www.kaushik.net/avinash/excellent-analytics-tip-11-measure-effectiveness-of-your-web-pages/*

Krug, S. (2006), *Don't Make Me Think!*, 2nd edition edn, New Riders.

Leech, J. (2013), *Psychology for Designers*, Five Simple Steps.

Mager, A. (2013), 'Way Out West Hack Battle 2'. Visited August 13, 2013.
URL: *http://devnews.spotify.com/2013/07/11/way-out-west-hack-battle-round-2/*

Mills, R. (2011), *A Practical Guide to Designing the Invisible*, Five Simple Steps.

Morton, J. (2010), 'Why Color Matters'. Visited November 8, 2013.
URL: *http://www.colorcom.com/research/why-color-matters*

Morville, P. and Rosenfeld, L. (2006), *Information Architecture for the World Wide Web*, 3rd edition edn, O'Reilly Media.

Net, U. (2006), 'Brainstorming'. Visited September 2, 2013.
URL: *http://www.usabilitynet.org/tools/brainstorming.htm*

Nielsen, D. L. (2004), Engaging Personas and Narrative Scenarios, PhD thesis, Copenhagen Business School.

Nielsen, J. (2001), 'Search: Visible and Simple'. Visited November 8, 2013.
URL: *http://www.nngroup.com/articles/search-visible-and-simple/*

Nielsen, J. (2002), 'Let Users Control Font Size'. Visited December 8, 2013.
URL: *http://www.nngroup.com/articles/let-users-control-font-size/*

Nielsen, J. (2006), 'How People Read on the Web: The Eyetracking Evidence'. Visited November 7, 2013.
URL: *http://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/*

Nielsen, J. (2010), 'Scrolling and Attention'. Visisted November 7, 2013.
URL: *http://www.nngroup.com/articles/scrolling-and-attention/*

Nielsen, J. and Loranger, H. (2006), *Prioritizing Web Usability*, New Riders.

Nielsen, J. and Tahir, M. (2001), *Homepage Usability: 50 Websites Deconstructed*, New Riders Publishing.

Norman, K. L. (2008), 'Better Design of Menu Selection Systems Through Cognitive Psychology and Human Factors', *Human Factors* **50**(3), 556–559.

Patrick Thibodeau (2013), 'India to overtake U.S. on number of developers by 2017'. Visited December 11, 2013.
URL: *http://www.computerworld.com/s/article/9240676/India_to_overtake_U.S._on_number_of_deve*

Pearson, C. (2011), 'Secret Symphony: The Ultimate Guide to Readable Web Typography'. Visited December 9, 2013.
URL: *http://www.pearsonified.com/2011/12/golden-ratio-typography.php*

Pernice, K. and Nielsen, J. (2009), How to Conduct Eyetracking Studies, Technical report, Nielsen Norman Group.

Preece, J., Rogers, Y. and Sharp, H. (2002), *Interaction Design: beyond human-computer interaction*, John Wiley & Sons, Inc.

Reichenstein, O. (2006), 'The 100Easy-2-Read Standard'. Visited December 8, 2013.
URL: *http://ia.net/blog/100e2r/*

Ross, J. (2009), 'Eyetracking: Is It Worth It?'. Visited November 4, 2013.
URL: *http://www.uxmatters.com/mt/archives/2009/10/eyetracking-is-it-worth-it.php*

Sajaniemi, J. (2008), 'Psychology of Programming: Looking into Programmers' Heads', *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments* **4**(4-8).

Sculley, D., Malkin, R. G., Basu, S. and Bayardo, R. J. (2009), Predicting bounce rates in sponsored search advertisements, *in* 'Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining', KDD '09, ACM, New York, NY, USA, pp. 1325–1334.
URL: *http://doi.acm.org/10.1145/1557019.1557161*

Snitker & Co (n.d.), 'Ten Steps to Personas'. Visited June 26, 2013.
URL: *http://personas.dk/wp-content/LOWRES-Personas-english-version-oktober-200821.pdf*

Sondalini, M. (n.d.), 'Understanding How to Use The 5-Whys for Root Cause Analysis'. Visited October 21, 2013.
URL: *http://www.lifetime-reliability.com/tutorials/lean-management-methods.html*

Suda, B. (2010), *A Practical Guide to Designing with Data*, Five Simple Steps.

Thompson, S. (2009), 'Putting personas under the microscope'. Visited December 5, 2013.
URL: *http://www.cooper.com/journal/2009/06/measuring_the_effectiveness_of*

Tidwell, J. (2011), *Designing Interfaces, Second Edition*, O'Reilly Media.

Travers, A. (2013), *A Pocket Guide to Interviewing for Research*, Five Simple Steps.

Travis, D. (2009), '247 web usability guidelines'. Visited June 24, 2013.
URL: *http://www.userfocus.co.uk/resources/guidelines.html*

Tullis, S. and Tullis, S. (2007), Statistical analyses of e-commerce websites: can a site be usable and beautiful?, *in* 'HCI International'.

Ware, C. (2004), *Information Visualization*, 2nd edition edn, Morgan Kaufmann.

A

Interviews

A.1 Interviews with Spotify employees

Note: *The interviews with the employees of Spotify have been excluded from this version of the report due to a non-disclosure agreements (NDA). For further details please contact the authors.*

A.2 Interviews with professional developers

Note: *The interviews with the professional developers have been excluded from this version of the report due to a non-disclosure agreements (NDA). For further details please contact the authors.*

A.3 Interviews with WoW hackathon participants

Note: *The interviews with the participants of Spotify's Way out West hackathon have been excluded from this version of the report due to a non-disclosure agreements (NDA). For further details please contact the authors.*

B

Analytics data from Spotify's developer site

Note: *The analytics for Spotify's developer site have been excluded from this version of the report due to a non-disclosure agreements (NDA). For further details please contact the authors.*

C

User personas

C.1 David

David
Senior Developer
34
Character Feature
• Social
• Technically experienced
• Physically active
• Reflective
• Eager to learn
Context
• Team work
• High standard equipment
• High tempo
"Visualizations are nice. It makes it so much easier to understand the overall picture."
"I hate being interrupted, it messes up my flow."
"A dev site is not something I ever visit if I already know what to do."
"Trying out new stuff is really exciting."
"It's my responsibility that I deliver qualitative solutions. And do it fast!"
"I read a lot. It's important to know what you're doing."
"We usually know the structures, it's mostly implementation details we look up."
A dev site needs
• Great documentation
• Structure description
• Thoroughness
David has been working at various software development companies for the last 7 years. He's currently employed at a medium sized company where he creates software for consumers. Daily he works in-house in a team of five. He is responsible for the front-end but has a rich experience in back-end as well. When faced with a problem he often turns to his colleagues for help.
In his spare time he does a lot of hiking and traveling with his girlfriend. He always takes his bike to work, no matter the weather. Apart from this he also has coding as a hobby. During the years, David has done many one-man projects to solve personal everday problems and learn new technologies. He's also collaborated in larger projects with friends.
David is very keen on being up-to-date with new tech releases that help him get even better at what he does. He likes to really understand and master technologies before he puts it into practice and commits to it. Time is of the essence and David prefers not to waste it on irrelevant distractions and dead ends.

C.2 Simon

Simon
Student
20
Character Feature
• Curious
• Very good with technology
• Impatient
• Targeted focus
• Heads first
Context
• Laptop with 13" screen
• From home or public environment
• Parallel deadlines
"You know what's really useful? Code snippets! It's really nice."
"If something doesn't work right away, I try something else instead."
"I found an example on GitHub. Why should I try to do something when someone else has already done it for me?"
"I used a tutorial. It was very helpful."
"I like to find and try out new technology. I do what I think is fun."
A dev site needs
• Code examples
• Search functionality
• Correct answers
• Showcases of what you can do
• Getting started
Simon attends his third year at university stuyding computer science. He has just moved out of his parents' home and now lives in a small rented student apartment. Simon is experienced within developing and knows several languages on a intermediate level. Not an expert within any single language but prefers Python because of its simplicity and ease of use.
As he's still in school he is used to deadlines, Simon does not have a great amount of resources, neither in hardware, money nor time. On his spare time he programs web sites for a little extra money.
Simon does not spend a lot of time thinking and preparing before jumping into action. He is relatively used to working in teams but likes to be alone when he needs to focus on his programming. He is sometimes percieved as lazy by others because he doesn't perform well when he's not interested. However when Simon is passionate about a problem he goes all in to solve it but does not spend all his time with one possible solution. Instead he tries out different options until one sticks.

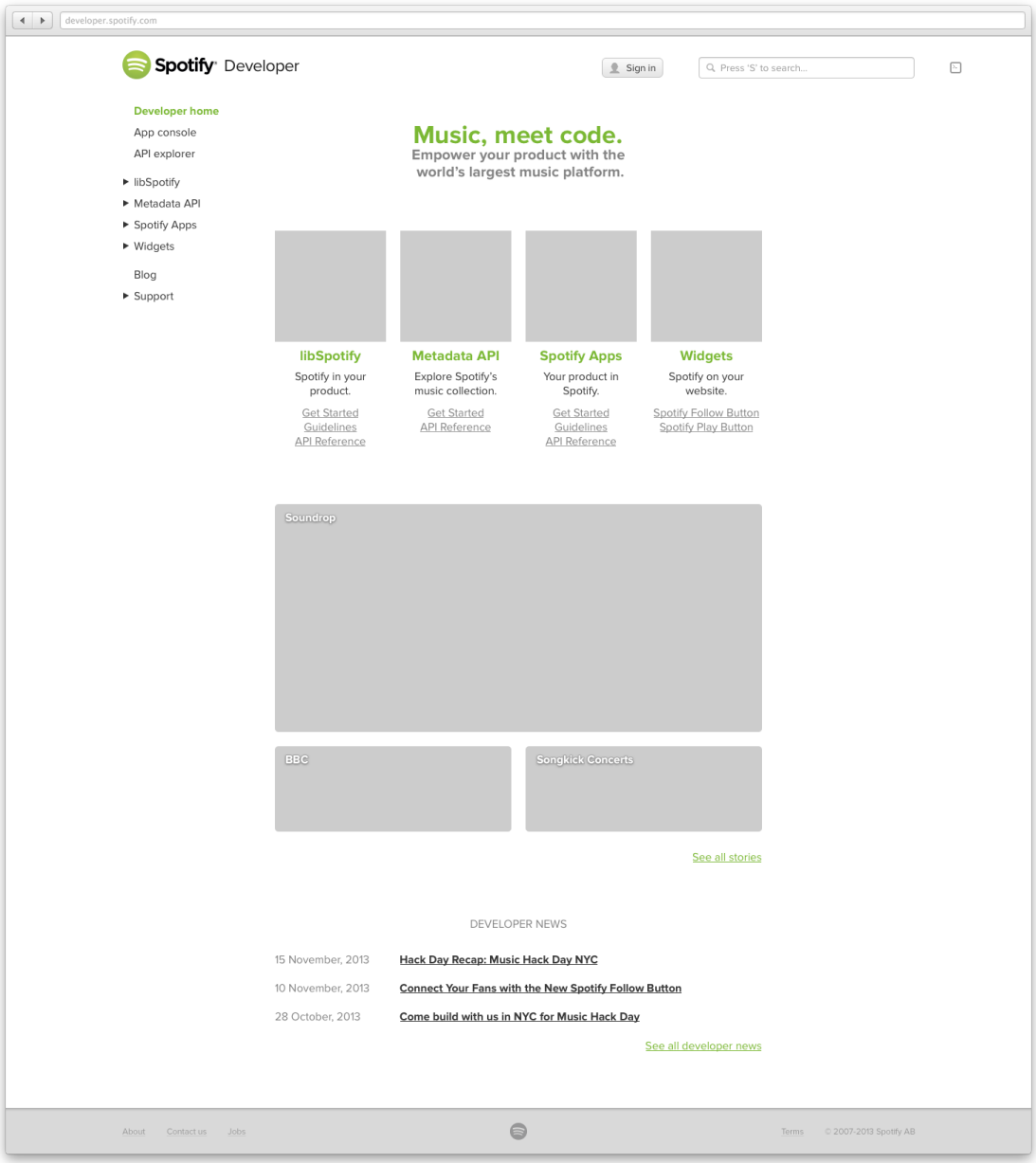
C.3 Tara

Tara
Tester/QA
28
Character Feature
• Social
• Structured
• Fast learner
• Service minded
Context
• Above average technology and design skills
• Team work
• Heavy work load
"I like helping people, but in an efficient manner."
"It wasn't really my responsibilty to fix it but since my client was dependent on a solution I fixed it anyways."
"Understanding why helps me explain to others better. Finding related information is key."
"I always have many simultaneous projects going on, so a good structure really helps."
Tara has been working as a software tester for 2 years. Before that she worked as a support agent at the support desk of a telecom company. She has a good amount of experience with technology but does not develop in her spare time. In addition Tara is very goal oriented, setting up goals and then working towards achieving them as she likes to finish what she has started.
Personally she loves to help out people out and sometimes places their needs before her own. This is in part because she loves to solve problems and also to see others grow. She gets on well with all kinds of people and has a gift for adapting to new situations. By questioning things she aims to always understand and learn more.
Tara likes to be efficient and has a morning routine but she has no problem changing it as long as there still is a routine. This helps her in her everyday life and job situation as she likes to have a lot of things to do but likes to save time. Furthermore she is a bit picky and values good quality in both her own and others' work.
A dev site needs
• Easy referencing
• Good explanations
• Quick look-up

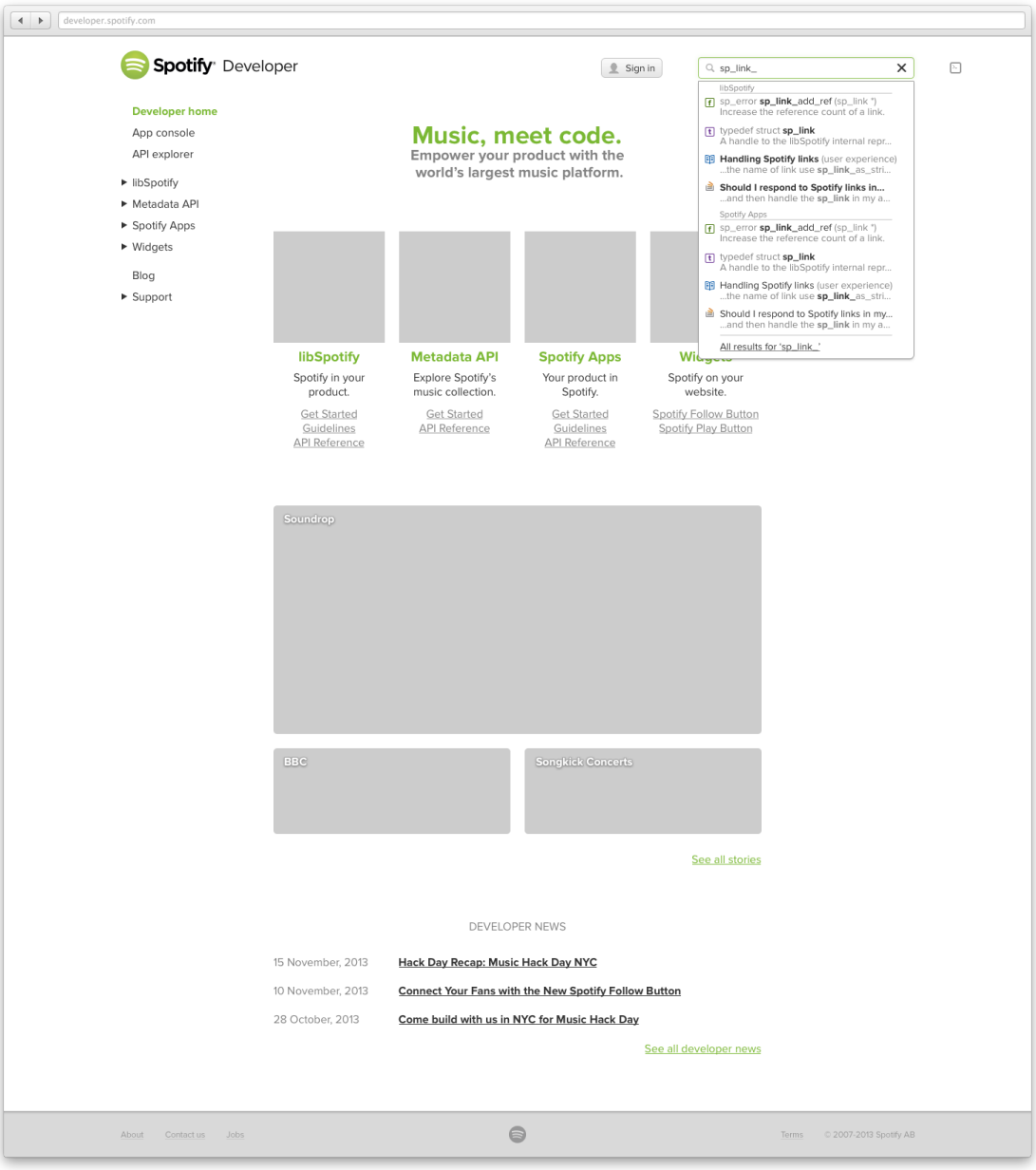
D

Prototype mockups

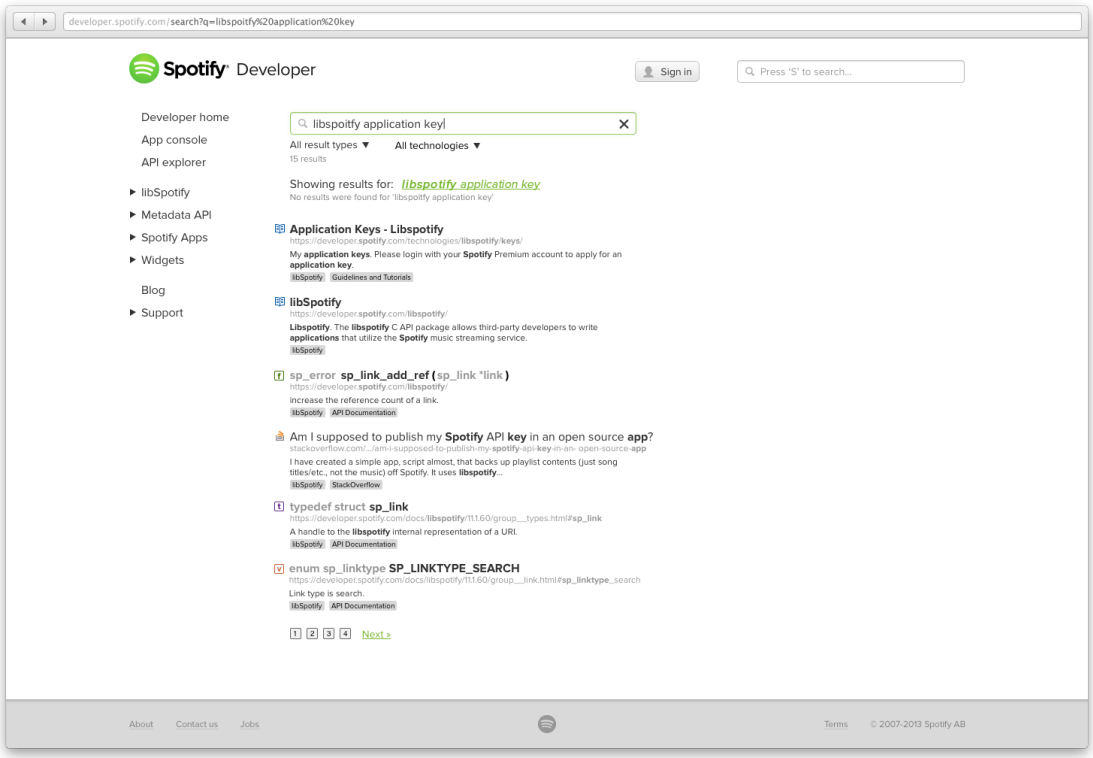
D.1 Home page


developer.spotify.com
Spotify Developer
Sign in
Press 'S' to search...
Developer home
App console
API explorer
libSpotify
Metadata API
Spotify Apps
Widgets
Blog
Support
Music, meet code.
Empower your product with the world's largest music platform.
libSpotify
Spotify in your product.
Get Started
Guidelines
API Reference
Metadata API
Explore Spotify's music collection.
Get Started
API Reference
Spotify Apps
Your product in Spotify.
Get Started
Guidelines
API Reference
Widgets
Spotify on your website.
Spotify Follow Button
Spotify Play Button
Soundrop
BBC
Songkick Concerts
See all stories
DEVELOPER NEWS
15 November, 2013
Hack Day Recap: Music Hack Day NYC
10 November, 2013
Connect Your Fans with the New Spotify Follow Button
28 October, 2013
Come build with us in NYC for Music Hack Day
See all developer news
About
Contact us
Jobs
Terms
© 2007-2013 Spotify AB

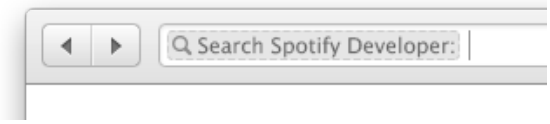
D.1.1 Home page with search popover



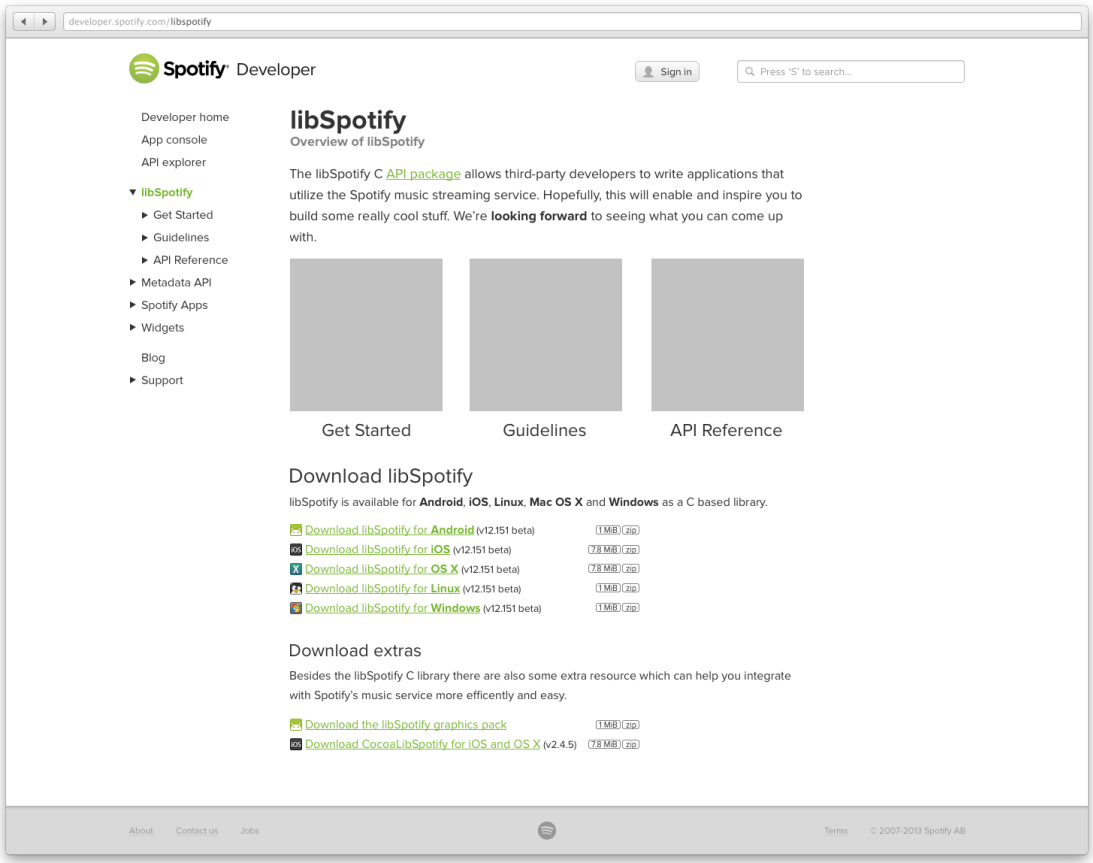
D.2 Search results



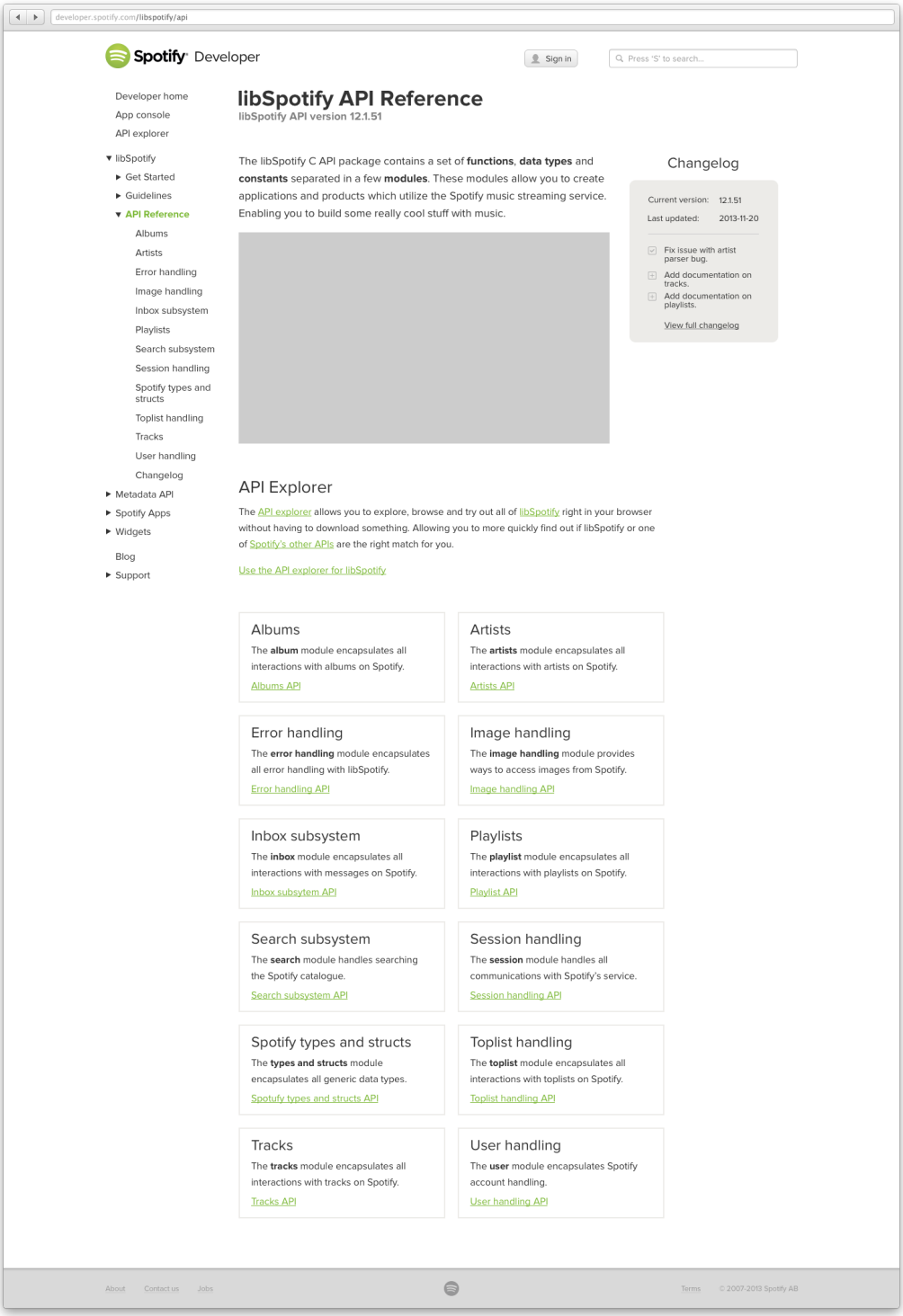
D.3 Search from the browser's address field



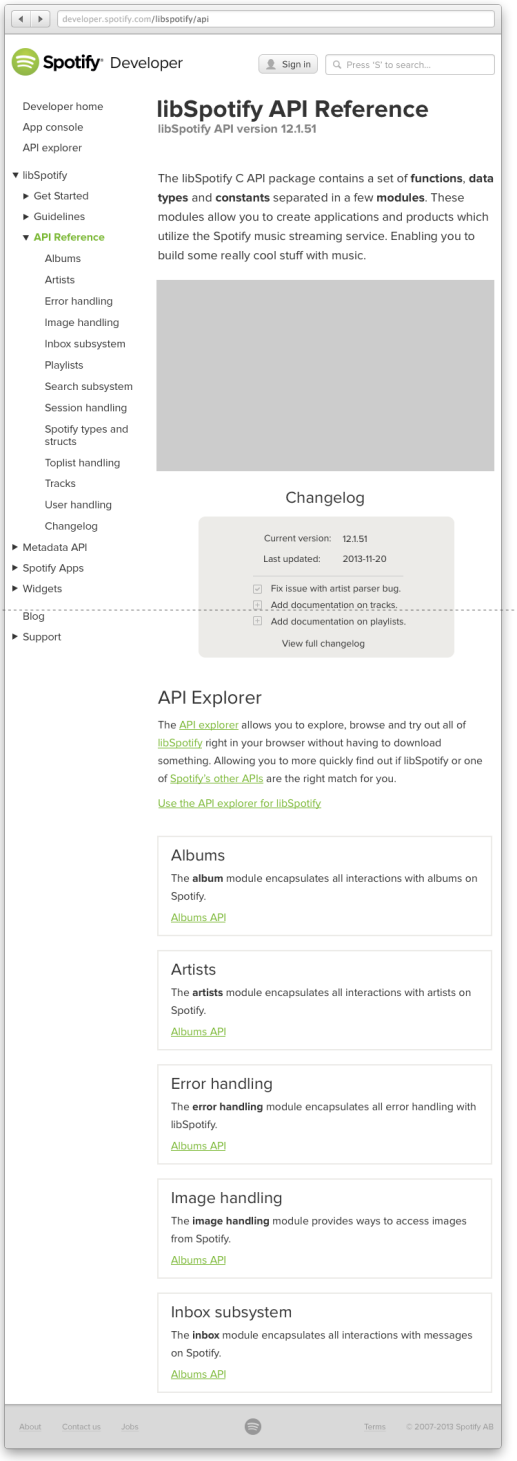
D.4 libSpotify



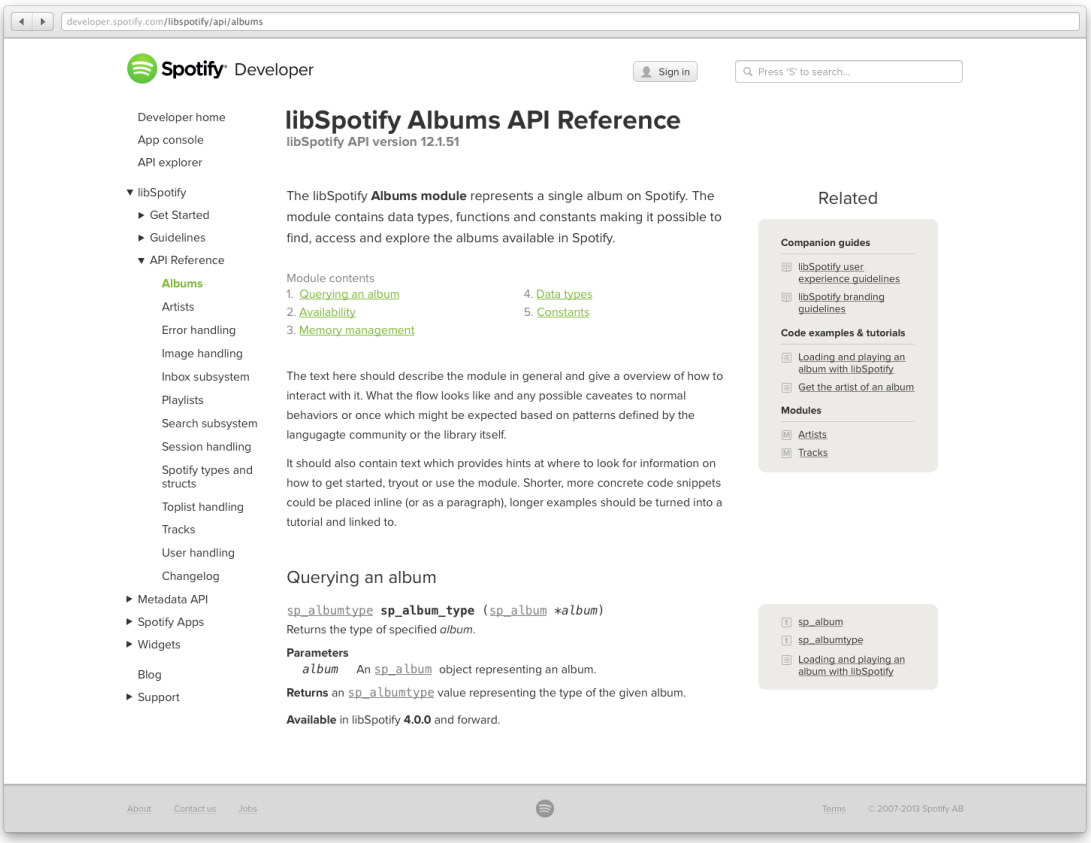
D.5 API reference



D.5.1 API reference halved



D.5.2 API reference for a single module



D.6 Power mode