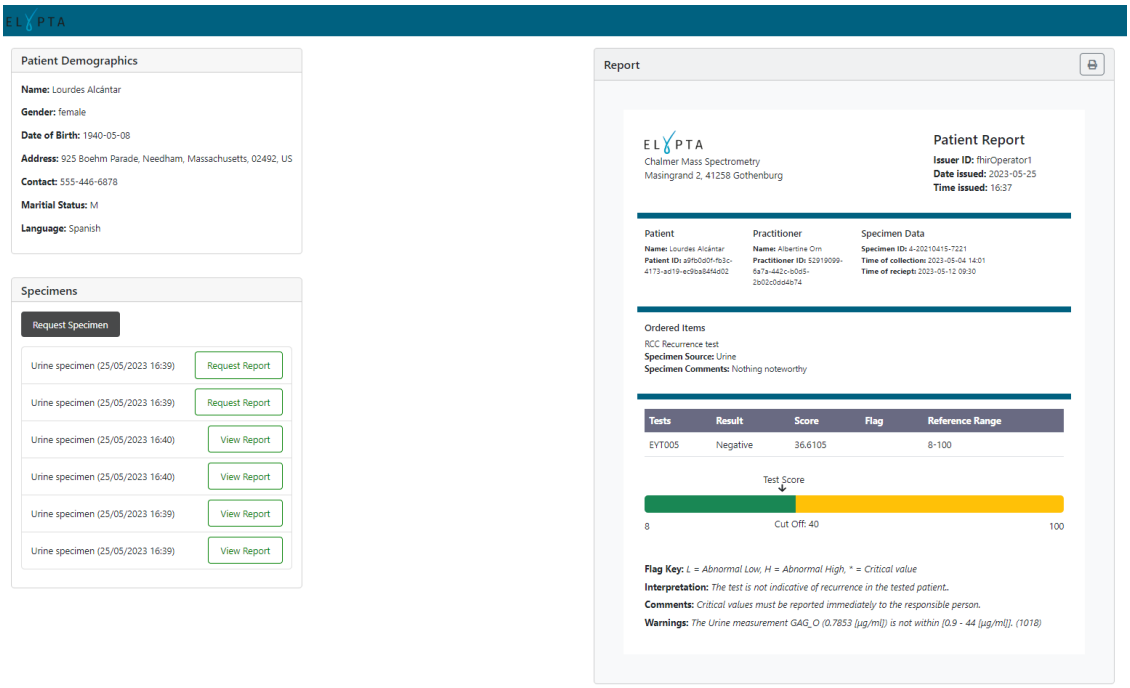


Teknikutvärdering och Proof of concept-utveckling SMART ON FHIR

En avhandling som utreder teknikens mognad i dagens EHR-system

Examensarbete inom högskoleingenjörsprogrammet Datateknik

Liam Mattsson

Viktor Rafstedt

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK

CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg 2023
www.chalmers.se

Examensarbete 2023

Teknikutvärdering och Proof of concept-utveckling SMART ON FHIR

En avhandling som utreder teknikens mognad i dagens EHR-system

Liam Mattsson
Viktor Rafstedt

GÖTEBORGS UNIVERSITET

CHALMERS

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
Göteborg 2023

Teknikutvärdering och Proof of concept-utveckling SMART ON FHIR
En avhandling som utreder teknikens mognad i dagens EHR-system
Liam Mattsson & Viktor Rafstedt

Handledare: Peter Olsson, TogetherTech och William Hughes, Göteborgs Universitet
Examinator: Jonas Duregård, Institutionen för Data- och Informationsteknik

Examensarbete 2023
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: Applikationens huvudsida.

Skriven i LaTeX
Göteborg 2023

Teknikutvärdering och Proof of concept-utveckling SMART ON FHIR
En avhandling som utreder teknikens mognad i dagens EHR-system
Liam Mattsson & Viktor Rafstedt
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet

Sammanfattning

I dagsläget är det välkänt att journalsystem bygger på gamla tekniker vilket gör de svåra att tekniskt arbeta mot. Vidare saknas det även standardiseringar mellan olika vårdgivare vilket gör det svårt att dela journaluppgifter mellan vårdgivare. Dock håller många journalsystem förnuvarande på att bli uppgraderade och en vanlig teknik i den implemenenteringen är FHIR (Fast Healthcare Interoperability Resources) som standardiserar journalinformation.

FHIR möjliggör även för skapandet av SMART on FHIR appar, vilka bygger på SMART specifikationen för hur appar kan byggas mot journalsystem i formen att de funkar mot samtliga journalsystem, oavsett leverantör. Det är en sådan applikation som medicinteknikföretaget Elypta, som möjliggjort detta examensarbete, efterfrågar i syfte av ett proof-of-concept. Elypta har tagit fram en teknik för hur man kan detektera cancer i biomarkörer från urin och blod.

Applikationen som utvecklats i detta examensarbete är applicerbar på samtliga framtida och nuvarande journalsystem som möjliggjort för SMART on FHIR applikationer. Applikationen möjliggör för en läkare att, inom sjukhusets egna journalsystem, kombinera journalsystemets samt Elyptas data för att beställa, visa och hantera tester och resultat från Elypta som en extern leverantör. Vidare har även en utredning genomförts kopplat till hur redo SMART on FHIR applikationer är att implementera inom den svenska sjukvården vilket visat att tekniken i sig är redo och förväntas nå en bredare implementation, dock är inte tekniken idag tillräckligt implementerad inom den svenska sjukvården för att SMART on FHIR applikationer skall vara användbara.

Nyckelord: SMART , FHIR, React.

Förord

Detta är ett examensarbete som ingår som ett obligatoriskt och examinerande moment i programmet “Högskoleingenjör Datateknik“ på Chalmers Tekniska Högskola. Examensarbetet omfattar 15 högskolepoäng och har utförts på halvtid från perioden Januari till Juni. Arbetet har utförts med samarbete mellan TogetherTech och Elypta AB, samt handledning från TogetherTech och Chalmers Tekniska Högskola. Idèn till examensarbetet kommer från Elypta AB som genom TogetherTech rekryterat studenter från Chalmers Tekniska Högskola att utföra detta.

Vi har flera personer att tacka för när det kommer till att slutföra detta examensarbete. Alla dessa personer har på ett eller annat sätt bidragit till att detta examensarbete blev verklighet och slutligen en bra produkt. Den första personen att tacka är våran handledare William Hughes vid Chalmers Tekniska Högskola som har handledit oss genom rapportskrivande och allt akademiskt arbete. Den andra personen vi vill tacka är Peter Olsson vid TogetherTech, som har gett oss handledning genom de tekniska bitarna av examensarbetet. Den tredje personen vi vill tacka är Gabriel Ortiz-Johansson, som har givit oss tydliga och bra instruktioner till hur slutprodukten skall fungera samt ställt upp på att svara på eventuella frågor om applikationen. Slutligen vill vi tacka Håkan Rolin, som varit med som ett stöd och haft en bra överblick på hur examensarbetet har gått, samt gjort så vi har trivts otroligt bra i TogetherTech:s lokaler.

Liam Mattsson & Viktor Rafstedt, Göteborg, Juni 2023

Beteckningar

Nedan är listan över beteckningar som har använts under hela detta examensarbete listade i alfabetisk ordning:

API	Application Programming Interface
APP	Applikation
CSS	Cascading Style Sheets
FHIR	Fast Healthcare Interoperability Resources
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
MVP	Minimum Viable Product

Innehåll

Akronymer **x**

Figurer **xv**

Tabeller **1**

1 Inledning **1**
1.1 Bakgrund . . . 1
1.2 Syfte . . . 1
1.3 Mål . . . 1
1.4 Avgränsningar . . . 2

2 Teknisk Bakgrund **3**
2.1 Git . . . 3
2.2 Gerrit . . . 3
2.3 JIRA . . . 3
2.4 FHIR . . . 4
2.5 SMART on FHIR . . . 4
2.6 HTML . . . 5
2.7 CSS . . . 5
2.8 JavaScript . . . 6
2.8.1 JSON . . . 6
2.8.2 React . . . 7
2.9 REST API . . . 7
2.9.1 JSON-Server . . . 7
2.10 Figma . . . 8

3 Metod **9**
3.1 Scrum . . . 9
3.1.1 Sprints . . . 10
3.1.2 User Stories . . . 10
3.2 Research . . . 11
3.3 Dokumentation . . . 11

4 Implementationen **13**
4.1 Applikationsutveckling . . . 13
4.2 Applikationsstruktur . . . 15

4.2.1 index . . . 15
4.2.2 App . . . 16
4.2.3 Header . . . 16
4.2.4 PatientDemographics . . . 16
4.2.5 ActiveRequests . . . 16
4.2.6 Tooltip . . . 17
4.2.7 Report . . . 17
4.2.8 Graph . . . 17
4.2.9 API . . . 17
4.3 Intervjustudie . . . 17
4.4 Kod-design . . . 18

5 Resultat **19**
5.1 Applikationsdemo . . . 19
5.2 Resultat från intervjustudie . . . 25

6 Diskussion **27**
6.1 Utvärdering . . . 27
6.1.1 Etiska perspektiv . . . 28
6.1.2 Hållbar utveckling . . . 28

7 Slutsats **29**

Bibliography **31**

Figurer

4.1 Strukturen över “Patient” resursen. 13
4.2 Prototypdesign för huvudsidan av applikationen. 14
4.3 Diagram över appens modeller. 15

5.1 SMART Launcher 19
5.2 SMART Login-sida 20
5.3 Sida där man väljer patienten 21
5.4 Huvudsidan 21
5.5 I denna ruta anges information om testet man begär från labbet. . . . 22
5.6 Listan över urin- eller blodprov som skapats. 22
5.7 Knapp för att visa rapport. 23
5.8 Den slutgiltiga rapporten. 24
5.9 Applikationen i sin helhet. 25

1
Inledning

1.1 Bakgrund

Detta examensarbete genomförs tillsammans med TogetherTech samt deras uppdragsgivare Elypta. TogetherTech är ett teknikkonsult företag med över 250 medarbetare som tidigare har haft Elypta som uppdragsgivare [1]. Elypta i sin tur är ett Stockholmsbaserat medicinteknikt företag som forskat och tagit fram en teknik för hur man kan upptäcka cancer på biomarkörer från blod och urin [2]. Elypta är för närvarande i stadiet då deras teknik genomgår kliniska tester och har därmed börjat utforska möjligheterna för mer spridda framtida lösningar redo för kund. En utav de potentiella lösningarna är att erbjuda sjukhus och kliniker tillgång till Elyptas plattform genom en så kallad SMART ON FHIR app. Detta möjliggör för en användare att inuti ett journalsystem ha simultan kommunikation med både Elyptas API SKY och journalsystemens egna data.

1.2 Syfte

Syftet med examensarbetet är att utvärdera SMART ON FHIR-teknologin och utveckla en proof-of-concept applikation inom området cancerdiagnostik, med hjälp av lämpliga ramverk och webbprogramvarutekniker. Detta görs för att kunna studera hur redo denna teknik är för användning inom svenska sjukhusmiljöer. Dagens applikationer körs ofta med gamla tekniker och system som kan innebära säkerhetsrisker, som till exempel svaga autentiseringsprocesser för tillgång till elektriska journaler. Därför behövs en applikation som är byggd på modernare teknik som är designad för att tackla dagens utmaningar kring användarvänlighet och säkerhet. Applikationen riktar sig främst till personer som arbetar på sjukhus, exempelvis läkare.

1.3 Mål

Målet med examensarbetet är att färdigställa en acceptabel prototyp av applikationen för att kunna bevisa att det är möjligt att utveckla klart en sådan applikation och att den kan användas i till exempel sjukhusmiljöer. Med acceptabel prototyp avses de krav som applikationen måste uppfylla, som vi fått av Elypta AB. Denna version kommer i denna rapport kallas för den simpla versionen. Om tid finns, så kommer en mer avancerad version av appen utvecklas med bland annat fler funktioner och datahämtning med hjälp av SMART ON FHIR. Applikationen ska kunna

köras genom ett journalsystem.

1.4 Avgränsningar

Flera avgränsningar introducerades tidigt i examensarbetet. En utav dessa är att applikationen ska vara gjord för att främst fungera på en dator. Tekniskt sätt fungerar även applikationen om man skulle starta den på en mobil enhet, men eftersom journalsystemen på sjukhus körs via datorer så var detta något som arbetet behövde förhålla sig till.

Eftersom detta endast är en proof-of-concept applikation, så är en utav avgränsningarna också att applikationen bara kommer att ha de mest nödvändiga funktioner som bevisar att den fungerar och svarar på frågan ifall denna teknik är redo att introduceras i svenska sjukhusmiljöer. Därav kommer funktioner som inte är specificerade av Elypta AB inte att implementeras, men också på grund utav tidsintervallet som examensarbetet befinner sig inom.

För applikationer utvecklade mot journalsystem blir alltid säkerhet en viktig aspekt. Dock kommer det inte vara fokuset under arbetet då det inte är efterfrågat i proof of concept stadiet från Elypta. Vidare hanteras även säkerhetsaspekterna främst av producenterna av journalsystemen samt sjukhusen som implementerar dem, vilket hade kunnat göra det relevant att ta upp i utvärderingen. Dock ansågs även en djupare analys inom den säkerheten ligga utanför ramarna för examensarbetet och kommer därav ges begränsad uppmärksamhet i rapporten.

2
Teknisk Bakgrund

I detta kapitel beskrivs tekniker, bibliotek samt programmeringsspråk som har använts i samband med examensarbetet.

2.1 Git

Git är ett gratis distribuerat versionskontrollsystem med öppen källkod designat för att hantera allt från små till mycket stora projekt med snabbhet och effektivitet[3]. Det fungerar igenom att en eller flera personer laddar upp sin kod till en huvudgren. Från huvudgrenen kan man sedan skapa andra grenar som är baserad på den, så att man kan fortsätta utveckla på olika områden inom applikationen. Sedan kan man genom en "pull-request" skicka in sin kod i huvudgrenen. En pull-request är som en förfrågan där man skickar upp sin färdiga kod för sina medarbetare att titta på samt testa. Ifall koden funkar som den skall så godkänner sedan personen detta och när det korrekta antalet personer har godkänt koden så skickas den in i huvudgrenen. Om personerna inte godkänner koden, så har man möjlighet att lämna synpunkter på koden. På det sättet så undviker man kod som krockar med varandra. En till fördel med Git är att man kan ladda ner koden som någon laddar upp, vilket gör det mycket lättare ifall man behöver byta enhet som man arbetar på.

2.2 Gerrit

Gerrit är ett verktyg för kodsamarbete som används som en Git-plattform samt för granskning av kod. Gerrit skiljer sig i användandet jämfört med till exempel Github, som är den mest populära Git-plattformen. Vanligtvis, så laddar man upp koden till olika grenar som man sedan slår samman med koden i huvudgrenen där koden för den fullständiga applikationen befinner sig. I Gerrit så får man inte ladda upp koden direkt till huvudgrenen. Först måste koden granskas av flera andra personer i utvecklingsteamet för att bli accepterat in i Git-repot.

2.3 JIRA

JIRA är en mjukvara utvecklad av Atlassian som erbjuder agila verktyg, framförallt för projkektledning.I JIRA har vi främst använt deras Scrum Boards funktion, men i JIRA kan man utöver det bland annat skapa workflows eller testa mjukvara[4].

2.4 FHIR

FHIR (Fast Healthcare Interoperability Resources) är en modern standard som definierar hur hälsoinformation byts ut mellan system. FHIR har utvecklats i syfte att möta det behovet som uppstått inom sjukvården för lättillgänlig och standardiserad patientdata. Behovet finns både inom de traditionella sjukvårdsområdena, men också från nya perifera industrier som till exempel med-tech och nätläkare där många applikationer är i utvecklingsstadiet vilka har behov av standardiserad patientdata. Standarden är framtagen av Health Level Seven International (HL7) [5] och i Sverige pågår det för nuvarande en statlig utredning (S 2022:10)[6] för hälsodata som nationellt intresse där FHIR troligtvis kommer att spela en stor roll. Det har även skett akademisk forskning kirng implementatonen av FHIR som visat att det främst i USA har börjat implementeras men att det funnits en del växtvärk i systemet och att det troligen kommer att behöva förbättras innan det blir välspritt [7][8].

Fördelen FHIR ger mot tidigare lösningar, som ofta var unikt utvecklade för sitt användarsyfte och som ofta inkluderade mycket handpålägg, är att det är mer flexibelt, skalbart och modulärt. Syftet med utvecklingen av FHIR var att bemöta den anstormade "applikationsekonomin" som även börjar märkas inom sjukvården, vilket kräver ett system som enkelt och lättviktigt kan utbyta data genom moderna internetteknologier för att skapa synergier mot redan existerande lösningar som skall implementeras inom sjukvården[5].

Som inom all sjukvårdsrelaterad datahantering är säkerhet en viktig aspekt, och detta blir speciellt relevant i fallet FHIR då det används för utbyte av patientdata. FHIR är anpassat för att vara kompatibelt med Health Insurance Portability and Accountability Act (HIPAA), vilket är en amerikansk lag som syftar till att reglera delning av känslig patientinformation. Vidare använder sig FHIR av 0Auth2-protokollet för alla applikationer som använder systemet. FHIR i sig är dock varken en hjälte eller en bov i diskussionen kring säkerhet då syftet med FHIR endast är att standardisera formatering av digital hälsodata. Dock öppnar FHIR upp för ett mycket bredare utbyte av hälsodata genom till exempel SMART ON FHIR applikationer. Denna ökade mängd av transfererad data samt den snabba utvecklingen orsakar dock säkerhetsrisker vilket behöver tas hänsyn till i utvecklingen av applikationer där FHIR standarderna används[9].

2.5 SMART on FHIR

SMART on FHIR började tas fram under 2009 då ett behov hade uppstått för en specifikation kring skapandet av appar inom journalsystem. Det främsta syftet med SMART on FHIR är att uppnå en standardisering vilket gör att en app utvecklad för ett journalsystem blir applicerbar i vilket journalsystem som helst. Vidare är det även ett sekundärt syfte att göra appar enkelt utbytbara vilket skapar en stark konkurrensutsättning som i teorin ökar kvaliten på applikationerna och sänker kostnaderna. Dock skapar det ökade utbytet, som tidigare diskuterat, en nyintroducerad

säkerhetsrisk i systemen, dock kommer detta inte tas i hänsyn till i detta projektet baserat på projektets avgränsningar. Det kan dock nämnas att ett av syftena med SMART on FHIR är att öka säkerheten för FHIR baserade applikationer, och forksning har visat att SMART on FHIR åtminstonde är ett steg i rätt riktning[10]. SMARTs koppling till FHIR kommer från att alla SMART applikationer kommunicerar genom FHIR standarder. SMART använder själva liknelsen till Android- eller iOS-plattformarna fast med journalsystem som grund för att beskriva sig själva[11].

Idag är det nästintill bestämt att SMART on FHIR kommer att vara den framtida standarden för utveckling av applikationer inom journalsystem och myndigheter då det klarlagts av instutitioner som till exempel The Office of the National Coordinator for Health Information Technology (ONC)"[11]. Vidare är det implementerat i välkända verktyg så som Azure samt att Apple har support för SMART on FHIR applikationer för att nyttja patientdata framtagen i deras hälsoapplikationer [12][13]. Dock saknar många journalsystem fullt stöd för SMART on FHIR applikationer idag. Exempelvis pågår det ett stort arbete inom Västra Götalands Regionen (VGR) för att implementera ett nytt journalsystem kallat Cerner Millenium. Cerner Millenium skall ha stöd för SMART on FHIR applikationer men implementeringen är grovt försenad och förväntas vara färdig först 2026[14]. Det är idag oklart huruvida VGR har stöd för SMART on FHIR applikationer i sina system[15].

2.6 HTML

HTML är ett så kallat märkspråk som används för att bygga webbsidor. Till skillnad från ett programmeringsspråk så använder man endast fördefinerade koder för att hantera text, bilder och länkar med mera. Variabler, conditions och annan logik som annars är vanligt i en programmeringsmiljö existerar inte i HTML. Man använder dessa fördefinerade koder för att informera webbläsaren om hur man som utvecklare vill att de olika elementen ska visas på webbsidan. Eftersom HTML är ett märkspråk, så kan man säga att HTML beskriver strukturen på en webbsida.

Vi använde HTML för att bygga en enkel prototyp av applikationen. Specifikt så användes det för att visa text på en webbsida med bland annat patientinformation och testinformation.

2.7 CSS

CSS är ett språk som beskriver strukturen och stilen på ett dokument med HTML-komponenter. CSS kan göra det mesta för att ändra utseendet på både sidan själv men också komponenterna på sidan[16]. Detta kan till exempel vara att ändra font på text, ändra färg på komponenter, flytta på komponenter eller lägga till animationer i dokumentet. För att HTML-dokumentet ska känna igen CSS koden så behöver man antingen länka sin CSS fil i HTML filen, eller skriva CSS direkt i HTML filen med hjälp av “style” komponenten. Man designar sitt dokument genom att skapa

olika block där man skriver sin CSS. Själva CSS koden är fördefinerade koder som man byter värden på. Dessa koder har ett standardvärde från början, som man kan få tillgång till genom att använda verktyget "Inspektera" i en webbläsare. Man kan med fördel lägga HTML-komponenter i en "div" komponent som man döper till valfritt namn, som man sedan kallar på i CSS filen för att byta stilen på flera saker samtidigt.

I detta examensarbete har vi använt CSS som ett komplement till React Bootstrap. React Bootstrap inkluderar färdiga komponenter man kan använda[17]. Där vi har velat ändra på dem, som till exempel att ändra storlek eller bakgrundsfärg, har vi använt oss av CSS.

2.8 JavaScript

JavaScript är ett programmeringsspråk som används flitigt inom webbutveckling. Språkets främsta användningsområde är att göra en webbsida mer interaktiv. JavaScript delar liknande språkkonstruktioner med vissa andra populära programmeringsspråk. Dessa språkkonstruktioner är till exempel klasser, statements, loopar, objekt med mera.

JavaScript är det språk vi främst använt i applikationen. I prototypen används JavaScript för att sköta det på webbsidan som inte är statiskt. Detta innefattar autentisering, hämta patient, hämta tester med mera. De tutorials som följts som använder FHIR API:n har också använt sig utav JavaScript, så därför var det naturligt att fortsätta använda just detta programmeringsspråk.

2.8.1 JSON

JSON är ett textbaserat format som utbyter data för att lagra information[18]. JSON är även giltig JavaScript-kod. JSON är smidigt att använda då det fungerar med de flesta programmeringsspråk, och det är uppbyggt så att det är enkelt för en människa att läsa datan. JSON fungerar så att man deklarerar objekt som man sedan kan hämta med sitt programspråk man kodar i. Objektet har sedan flera nyckel-värde-par där nyckeln skrivs inom citattecken. Värdet kan representeras i flera former som till exempel textsträngar, tal eller null[19]. Detta kan till exempel se ut såhär:

{ "patients": [{ "name" : "Patrik Eriksson", "age" : 42, "gender" : "Male" }] }

Under utvecklingen av applikationen har vi erhållit testrapporter från Elypta AB i form av JSON filer. Detta för att vi ska kunna testa ifall datan visas korrekt i applikationen på en webbsida. Vidare används JSON i applikationen för att kommunicera med ELYPTA SKY genom att skicka en GET request till en rapport uppladdad på plattformen. När vi skickar vår request får vi tillbaka JSON-kod som vi sedan kan hantera i vår applikation.

2.8.2 React

React är ett JavaScript-bibliotek som används för att skapa interaktiva användargränsnitt inom webbutveckling[20]. Detta bibliotek är mycket populärt bland frontendutvecklare[21]. React är skapat av Meta som är väl kända för att tillhandahålla applikationer såsom Facebook och Instagram[22]. React underlättar skapandet av en webapplikation eftersom man delar upp webbsidan i olika komponenter som man skapar, för att sedan lägga ihop alla komponenter till en huvudsida. Biblioteket är open-source vilket öppnar upp för andra utvecklare att skapa sina egna bibliotek med React och sedan dela med sig. Detta betyder att det finns mycket användbara verktyg att använda med hjälp av React när man bygger en webbapplikation. Ett utav dessa bibliotek vi har använt är React Bootstrap, som är ett bibliotek med färdiga komponenter[17]. Eftersom React är ett JavaScript bibliotek så fungerar det att skriva vanlig JavaScript när man programmerar, men det finns också likheter med HTML då man kan använda samma komponenter i React. Detta kan till exempel vara rubriker, paragrafer eller inputfält.

2.9 REST API

En REST-API är ett API som följer satta begränsningar av ett ramverk kallat REST. Kort kan man säga att ett API är en kommunikationväg mellan två parter där det finns uppsatta regler för hur kommunikation skall genomföras. Ett exempel hade kunnat vara en kart-applikation där användaren kommunicerar sin nuvarande position och positionen hen vill till mot applikationens API, efter vilket API:n svarar med bästa transportvägen.

En REST API är ett API som har följt vissa riktlinjer i dess utveckling. De riktlinjerna är att 1) Klient-Server: det skall finnas en tydlig uppdelning mellan klient- och serversidan 2) Stateless: serversidan skall inte kunna lagra information om klienten. Vilket medför att varje request sker oberoende av tidigare requests. 3) Cachable: data skall kunna cachas för att minska serverbelastning 4) Uniform Interface: API:n skall vara konsekvent och enhetlig i termer av HTTP-metoder för att manipulera resurser. 5) Layered Systems: API:n skall vara uppbygd i lager i syfte av att tillåta enkel skalbarhet samt återanvändning av komponenter.[23]

2.9.1 JSON-Server

JSON-server är en Node.js implementation som möjliggör för skapandet av lokala "fake" REST APIer, där "fake" syftar på att det inte är ett verkligt API utan enbart en lokal JSON-implementering som simulerar ett verkligt REST-API. En vanlig beteckning för det är en "mock database". JSON-Server används vanligtvis i utvecklingmiljöer där man av diverse anledningar ännu inte vill kommunicera med en verklig REST-API utan istället laddar upp data på en lokal JSON-Server. [24]

2.10 Figma

Figma är en mjukvara som används för att designa sina egna användargränssnitt. Det som gör denna mjukvara attraktiv är att flera personer kan jobba på samma design i realtid[25]. Figma genererar även CSS kod tillhörande designen man skapar, vilket gör att man kan använda samma CSS i sin egna applikation. I vårt fall användes Figma till att designa huvudsidan till våran applikation.

3
Metod

3.1 Scrum

Under examensarbetet har vi använt oss av det agila ramverket Scrum. Detta används inom projektledning för att få en att effektivisera arbetet[26]. Scrum baseras främst på att man har en typ av elektronisk anslagstavla där man lägger upp tasks och user stories. Ett task är precis som det låter, nämligen en uppgift som ska göras. En user story är något som flera tasks bidrar till, det kan vara något beskrivande som till exempel en kund har sagt. Detta kan till exempel vara "Jag vill kunna skriva in mina uppgifter". Tasks och user stories delas upp i något som kallas Sprints, vilket är en tidsperiod som bestäms inom utvecklingsteamet. Detta kan vara allt från en vecka till en månad, men alla sprints har en och samma längd. Inom detta examensarbete så är sprintlängden en vecka. Inom denna tidsperiod så skapar man så många tasks och user stories som teamet känner sig bekväma med, och målet är att bli klara med alla dessa inom den specifika sprinten.

Inom Scrum finns tre roller som tilldelas de som arbetar med projektet. Dessa är Scrum Master, produktägare samt utvecklare. Scrum Mastern har en av de viktigaste rollerna i ett Scrum projekt. Personen med denna roll leder teamet under varje pågående sprint. Denna roll brukar också kallas för projektledare. Det är också Scrum Mastern som sköter den kontinuerliga kommunikationen mellan teamet och produktägaren. Scrum Mastern i detta examensarbete är Peter Olsson. Produktägaren är personen som ska ta emot produkten när den är klar. Produktägaren här är Elypta AB. Denna personen bestämmer user stories och tasks samt bestämmer en MVP vilket är en bestämd del av utvecklingen där man har bestämt att produkten är klar, men med utrymme att vidareutvecklas. Utvecklarna är de som jobbar med produkten som skall framställas. Personerna som har denna rollen har ansvaret att leverera de user stories och tasks som framförs av produktägaren under en Sprint.

Varje Sprint delas upp i tre delar: planering, utveckling och sist en utvärdering, som också kallas för retrospective. Under planeringen träffas produktägaren, projektets Scrum Master och utvecklingsteamet och pratar om vad som skall utföras under den kommande sprinten. Under detta möte bestämmer produktägaren hur mycket som ska göras under sprinten med hjälp av utvecklingsteamet som berättar hur mycket de kommer att hinna med under den utsatta tiden. Scrum Mastern utvärderar sedan detta beslut och uttalar sig ifall arbetet blir för stort för den sprinten. Under utvecklingen så påbörjar utvecklare sitt arbete på alla tasks och user stories

som bestämts under planeringen. I slutet av utvecklingsskedet så är målet att kunna visa arbetet man gjort samt att alla tasks och user stories ska vara klara. När en sprint är avklarad så görs en sprintutvärdering där alla inblandade får ge sin åsikt på hur sprinten gick. Detta används sedan i nästa sprint för att få arbetet att utvecklas i rätt riktning.

3.1.1 Sprints

För att hantera våra sprints så har vi använt oss utav Boardsfunktionen i JIRA. På grund av avsaknaden av rättigheter, så har vi inte möjlighet att starta sprints i JIRA, men vi kan däremot hantera tasks och user stories. Därför uppdaterar vi våra boards varje vecka för att simulera att en sprint är utförd. I Boards kan man också tilldela tasks till olika personer, detta gjorde att den personen som tilldelades ett task hade ansvar för att det skulle vara gjort till slutet av sprinten. För att verifiera att vår kod fungerar som den ska så har vi använt Gerrit:s "Code-review" funktion, som låter en användare se över sin egen eller någon annans kod innan den laddas upp i projektgruppens repository.

3.1.2 User Stories

Grunden till projektet ligger i två stycken user stories, en enklare och en mer avancerad. Som nämnt i sektion 1.3 så är målet för projektet att som minst utveckla den enkla versionen men även den avancerade versionen om möjligheten finns. Därav är arbetets metod att starta med utvecklingen av den enkla versionen för att sedan expandera till den avancerade versionen.

User story enkel version:
Som läkare vill jag kunna starta en SMART on FHIR applikation som jag authentisenterar mig emot genom vårat journalsystem. Vidare vill jag kunna välja en patient ur vårat patientregister där jag sedan kan visa resultatet av ett test som gjorts. I resultaten vill jag se nödvändiga demografiska data om patienten samt de nödvändiga delarna av testresultatet så som om cancern är i remmision eller ej samt hur man skall tolka resultatet.

User story avancerad version:
Som läkare vill jag kunna starta en SMART on FHIR applikation som jag authentisenterar mig emot genom vårat journalsystem. Vidare vill jag kunna välja en patient ur vårat patientregister där jag sedan kan välja bland gjorda tester på patienten, jag vill även kunna se om testerna är under analys eller om dom är färdiga, och sist kunna välja ett prov för att antingen beställa eller visa resultatet. I resultaten vill jag se nödvändiga demografiska data om patienten samt de nödvändiga delarna av testresultatet så som om cancern är i remmision eller ej samt hur man skall tolka resultatet.

3.2 Research

Liam hade tidigare erfarenhet av HTML, CSS, JavaScript och React, men ingen utav oss hade tidigare erfarenhet av att jobba i en sandboxmiljö med hjälp av en API så som SMART ON FHIR. Detta medförde att vi behövde söka information om detta API, samt ta del av den dokumentation av SKY samt SMART ON FHIR som tillhandahålldes av Elypta AB, som tidigare hade arbetat med detta. För att bli bekanta med både SKY och SMART ON FHIR så läste vi igenom mjukvaruimplementationen som hade gjorts utav TogetherTech, samt TogetherTech:s mjukvarupolicy. Trots detta, så var det inte tillräckligt med information för att börja utvecklandet av applikationen.

Detta gjorde att ett extramöte med Elypta AB bokades, där tydliga instruktioner angavs vad som krävs för att examensarbetet ska kunna godkännas, samt i vilken ordning arbetet rekommenderades att arbeta i. Gabriel från Elypta rekommenderade också att vi skulle utföra 2 stycken tutorials på SMART ON FHIR. En utav dessa tutorials hade vi jobbat med innan, nämligen Cerner:s tutorial[14]. Han rekommenderade också att vi skulle använda SMARTs egna tutorial som komplement till Cerners i research syfte[27].

3.3 Dokumentation

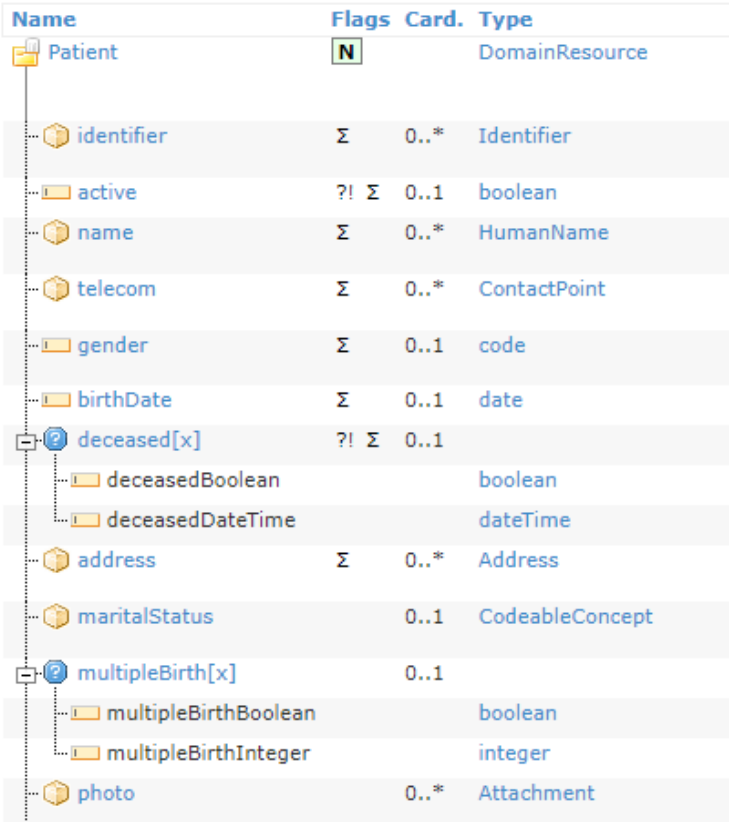
I utvecklingsskedet har främst två metoder använts för att dokumentera arbetet. Dokumentation har gjorts för flera anledningar. En av anledningarna är att som utvecklare kunna gå tillbaka till arbetet och snabbt förstå vad en bit kod gör för något utan att behöva läsa igenom hela koden. En annan anledning är för eventuellt framtida utveckling av applikationen. Om det finns dokumentation kring applikationen blir det lättare för den som tar över arbetet att fortsätta där man senast avslutade. Den första metoden vi har använt för dokumentation är kommentarer i koden. Det fungerar som dokumentation för utvecklarna själva. Den andra metoden vi har använt är extern dokumentation, där vi har använt textskrivningsprogrammet Overleaf för att varje vecka skriva vad vi har gjort för arbete under den veckan. Detta är samma program som används för att skriva rapporten på arbetet. Vi använder denna metod för att kunna läsa igenom vad som gjorts i tidigare skeden i utvecklingsprocessen, och på så sätt kunna beskriva genomförandet av arbetet bättre i denna rapport.

4

Implementationen

4.1 Applikationsutveckling

Vi började utvecklingsprocessen med att följa olika tutorials kring hur man skapar en SMART on FHIR applikation. Dessa tutorials byggde på att bygga en statisk sida med HTML för att kunna visa patientinfo hämtad från en FHIR server. Eftersom vi ville visa specifik info om patienterna, så använde vi oss av Patientresursen, som delar av kan ses i Figur 4.1. Denna resurs är en del av FHIR och inkluderar attribut som till exempel namn, kön och födelsedatum. Samtliga fhir resurser och hela patient resurser finns i FHIR dokumentationen[28].

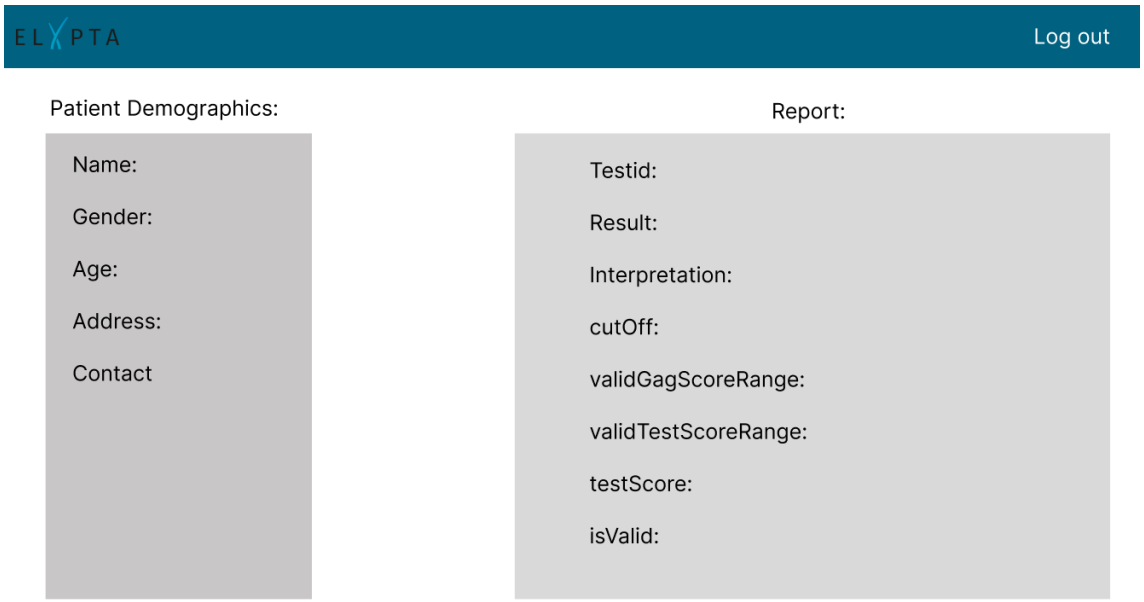


Name	Flags	Card.	Type
Patient	N		DomainResource
identifier	Σ	0..*	Identifier
active	?! Σ	0..1	boolean
name	Σ	0..*	HumanName
telecom	Σ	0..*	ContactPoint
gender	Σ	0..1	code
birthDate	Σ	0..1	date
deceased[x]	?! Σ	0..1	
deceasedBoolean			boolean
deceasedDateTime			dateTime
address	Σ	0..*	Address
maritalStatus		0..1	CodeableConcept
multipleBirth[x]		0..1	
multipleBirthBoolean			boolean
multipleBirthInteger			integer
photo		0..*	Attachment

Figur 4.1: Strukturen över "Patient" resursen.

Innan vi började koda vår React-sida, så designade vi en enkel huvudsida med hjälp av mjukvaran Figma[29]. Målet var att försöka få den att se så enkel ut som

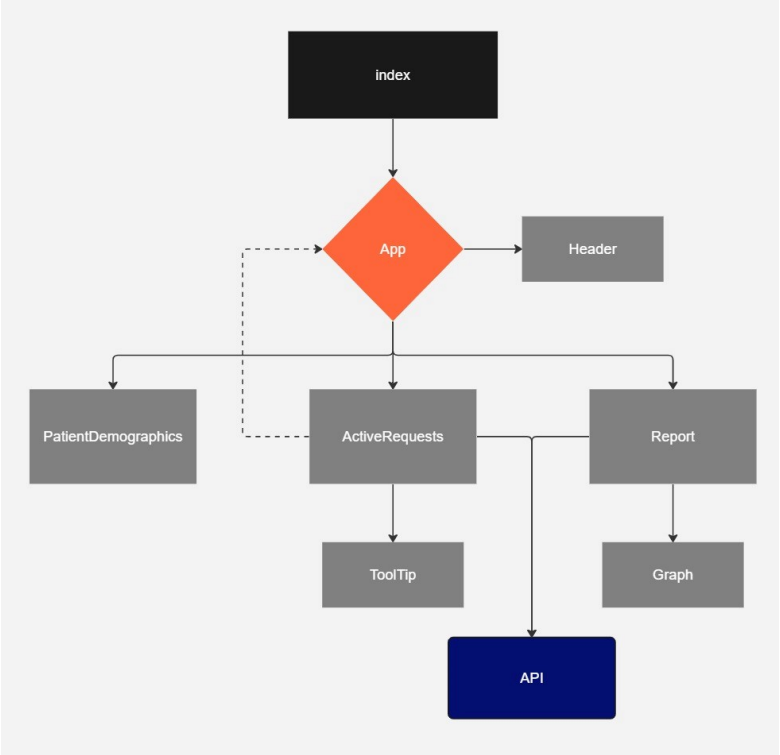
möjligt, så att även en med mindre teknisk kunskap skulle kunna använda den, resultatet finns i figuren nedan. Sidan vi designade skulle efterlikna funktionaliteten som den slutgiltiga applikationen skulle ha, men designen skulle endast fungera som en prototyp för stunden.



Figur 4.2: Prototypdesign för huvudsidan av applikationen.

Under "Patient Demographics" finns allmän info om patienten man har hämtat från FHIR servern, såsom namn, kön, adress med mera. Under "Report" finns det information om själva testet på patienten. FHIR tillhandahåller som visat tidigare i Figur 4.1 information om en resurs, så dessa var endast ett urval av de viktigaste resurserna att visa på sidan. Testid visar vilket id testet har, varje test har ett unikt testid som består av bokstäver, siffror och symboler. Result visar ett booleskt värde, som anger om patienten har cancer eller inte. Interpretation är en fri textrad där en läkare kan välja att lägga en kommentar på testet om så behövs. cutOff är ett tal som vid standard är 40, om testScore är större än det talet så har patienten cancer, men om det är mindre än det talet så har patienten inte cancer. ValidTestScoreRange och ValidGagScoreTestRange är ett intervall mellan två tal som visar vilket intervall där ett test anses vara giltigt. testScore är testets resultat, testScore styr tidigare nämda "Result" där alla värden över cutOff-värdet genererar positivt resultat och alla värden under cutOff-värdet genererar negativt resultat. isValid visar ett booleskt värde huruvida testet är giltigt eller ej. Vidare finns ett sidhuvud med Elyptas logga och en knapp där man kan logga ut från applikationen.

4.2 Applikationsstruktur



Figur 4.3: Diagram över appens modeller.

Figuren ovan visar strukturen på applikationen. I stora drag kan det beskrivas som att index, som initieras genom en SMART ON FHIR launcher, initierar appen som i sin tur initierar de olika komponenterna Header, PatientDemographics, ActiveRequests och Report. ActiveRequests initierar i sin tur ToolTip och Report initierar i sin tur Graph.

I flödet kommunicerar även ActiveRequest och Report med en API som i sin tur kommunicerar tillbaka data till ActiveRequests och Report. ActiveRequests kan även kommunicera tillbaka till appen då det skall göras ändringar i Report.

4.2.1 index

Index är ingångsporten till applikationen och uttnytjar FHIR-klientbiblioteket för att autensitisera sig mot journalsystemetet via 0Auth2. Autentiseringen kräver i vissa journalsystem ett bestämt client-id vilket index innehåller. Detta client-id är unikt till applikationen. Vidare bestämmer index vilket scope som applikationen ska ha tillgång till, vilka läs- och skrivrättigheter applikationen har mot journalsystemet. Blir autentiseringen och scopen accepterade startas smart applikationen genom App som får en client som returneras av index som prop.

4.2.2 App

"App" är huvudkomponenten i den här applikationen och agerar som en container för flera underkomponenter. Den tar emot ett klientobjekt som en prop, som används för att interagera med FHIR-tjänsten.
"App" är applikationens huvudkomponent, eller parent, och agerar som en container för flera underkomponenter. Den tar emot ett client object från index som den sedan använder för att kommunicera med journalsystemet. Utöver att rendera och kontrollera de andra komponenterna innehåller även app en metod som används utav ActiveRequests för att uppdatera vilken rapport som skall visas.
Slutligen returnerar App de olika komponenterna i kombination med den datan de olika komponenterna behöver, där client är det viktigaste som behövs i samtliga komponenter förutom header.

4.2.3 Header

Header är den enklaste komponenten. Det är en statisk komponent utan någon annan funktion än att visa företagets logotyp samt att bidra i estetiskt syfte.

4.2.4 PatientDemographics

PatientDemographics syftar till att samla in och formattera all önskvärd patientdata ur journalsystemet samt hur det visuellt skall presenteras. Den använder client objektet den fått som prompt av app och identifierar därigenom den aktiva patienten. PateintDemographics visar patientens namn, kön, födelsedatum, adress, telefonnummer, civilstånd samt språkkunskaper.

4.2.5 ActiveRequests

ActiveRequests kontrollerar alla prover som existerar på patienten samt ger möjligheten till att beställa nya prover. Samtliga beställda samt analyserade prover på en patient visas i en lista. Komponenten använder sig huvudsakligen av två stycken FHIR resurser, specimen och diagnostic report. Flödet på en ej tidigare använd patient är först att man begär att ett specimen, vilket i denna applikations fall är ett urinprov, som skall skapas på patienten. I ett verkligt scenario hade en patient lämnat ett urinprov som sedan hade analyserats och resultatet hade skickats till Elypta för vidare analys. Dock är det flödet varken nödvändigt eller möjligt i applikationen och istället skapas ett tomt specimen med en koppling mot ett redan analyserat specimen i SKY.
Detta skapade specimen har initialt statusen "unavailable" då det inte finns något resultat i SKY ännu. När läkaren begär en rapport på specimen skickas en request till SKY att specimenet skall analyseras, efter vilket det finns ett resultat i SKY. Det resultatet hämtas av applikationen i form av en diagnostic report, som sparas mot journalsystemet med en koppling till det specimen som resultatet har sitt ursprung från. När diagnostic reporten är sparad i journalsystemet kan användaren välja att visa rapporten i applikationen.

4.2.6 Tooltip

Då det är svårt att namnge de diverse knapparna så att det går att förstå vad knapparna gör existerar det tooltip för samtliga knappar i ActiveRequests. Tooltipsen visas då användaren har hållt muspekaren över en knapp i 1500ms och visas då som en pop-up över knappen där det står vad knappens funktion är.

4.2.7 Report

Reports funktion är att visualisera rapporten på ett både informativt och visuellt tydligt sätt. Report är i sitt ursprungsläge dold på skärmen, men när knappen "View Report" ur listan ActiveRequests trycks så kommer Report-komponenten att hämta datan ur SKY för det givna provet och visa en rapport i applikationen. Rapporten använder sig även av ytterligare en komponent, Graph, för att visualisera resultatet. Report har även en inbyggd möjlighet för att möjliggöra utskrift av rapporten där den inbyggda utskriftsfunktionen i webbläsaren används för att skapa en utskrift av hela Report komponenten.

4.2.8 Graph

Graph är en manipulerad React timeline där det minsta möjliga, högsta möjliga samt "Cut Off" värdet för testet visas. Cut Off är i sammanhanget det värde som bedömer om testet är negativt eller positivt baserat på om testvärdet är lägre eller högre än Cut Off värdet. Till sist visas även en pil över grafen som indikerar vart den nuvarande patientens testvärde ligger.

4.2.9 API

API komponenten används för all kommunikation med SKY som utnyttjas i ActiveRequests och Report. Det finns två möjliga funktioner att utnyttja i APIn, den ena är "postData" som postar en ServiceRequest till SKY, vilket innebär att man begär att SKY skall analysera ett taget test. Vidare finns även metoden "getData" som hämtar resultatet av ett prov givet ett id.

4.3 Intervjustudie

Frågeställningen som vi sökt besvara är hur mogen FHIR och dess kringliggande verktyg är för utveckling av medicinska appar. Denna intervjustudie kommer att baseras på vår egen erfarenhet av att ha utvecklat en sådan applikation, men även baserat på information från individer inom medicinteknik. Personen vi har kommunicerat med under denna intervjustudie är Petter Wolff från Sahlgrenska Science Park.

När vi sökte efter en person som är både insatt i FHIR samt hur applikationer i sjukhusens journalsystem ska fungera så dök Petter upp. Bland de personer vi fått fram så verkar han vara en utav de som har störst erfarenhet och kunskap vad gäller FHIR i Sverige genom sin position som business advisor på Sahlgrenska Science

Park samt utredare i statlig utredning (S 2022:10) som kretsar kring standardiserad digitalisering av den svenska sjukvården [6][30].

4.4 Kod-design

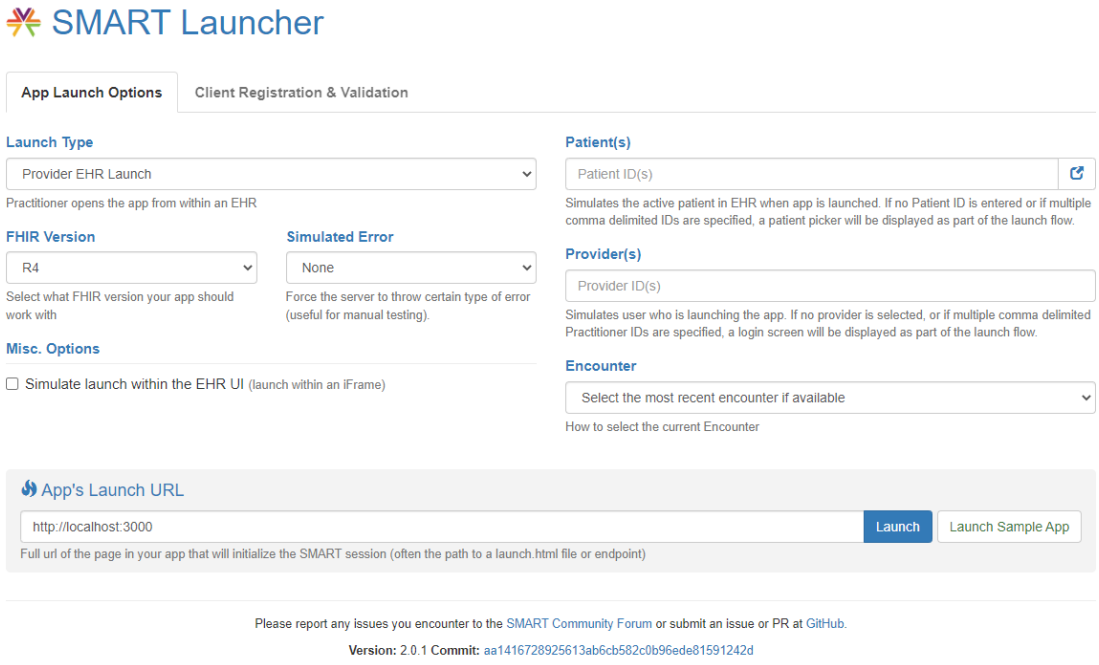
Vi har förhållit oss till olika tillvägagångssätt under utvecklingsprocessen. Innan utvecklandet, så studerade vi diverse design- och projektdokument som bland annat beskriver SKY:s REST-API och designval. Vi gjorde sedan tutorials för att bli bekanta med verktyget SMART on FHIR. Gällande utvecklandet i React så har vi delat upp koden i olika komponenter enligt deras egna programmeringsregler. Vi har även använt oss utav “useStates“, som gör att vi kan lägga till tillståndsvariabler till våra komponenter. Dessa i sin tur fungerar som ett tillägg till vanliga variabler då React inte tar hänsyn till om vanliga variabler har ändrat sitt tillstånd. Dessa tillståndsvariabler används till exempel till att uppdatera sidan när man klickar på en knapp. Vi har även följt React:s namnkonventioner när det gäller att namnge filer och kompontenter, nämligen att namn på alla komponenter ska börja med en stor bokstav.

5

Resultat

5.1 Applikationsdemo

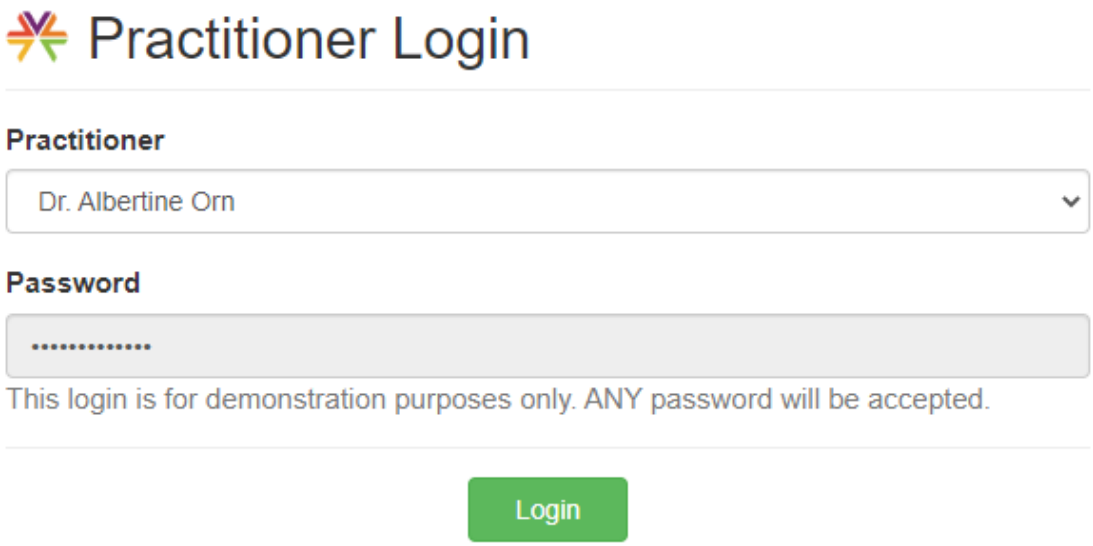
Utvecklingsprocessen varade under åtta sprints, vilket enligt tidsplanen över examensarbetet var precis tillräckligt för att ha tid att färdigställa rapporten och göra testning på applikationen. Denna del av avhandlingen kommer att visa det slutgiltiga resultatet av applikationens utseende och funktionalitet. Detta applikationsdemo kommer att demonstreras genom att använda SMART:s egna sandboxmiljö, som ska simulera ett riktigt journalsystem. Man startar applikationen genom SMART:s "SMART Launcher", som används för att starta applikationer som skapats med SMART ON FHIR teknologin. I SMART Launcher kan man välja att starta applikationen med flera olika alternativ. Detta är till exempel olika FHIR versioner, simulerat fel, eller genom ett simulerat användargränssnitt för journalsystem. Innan vi kör applikationen i SMART Launcher så behöver vi först starta appen lokalt på en valfri ledig port. I vårt fall väljer vi port 3000, så därför skrivs "http://localhost:3000" in i fältet där det står "App's Launch URL".



Figur 5.1: SMART Launcher

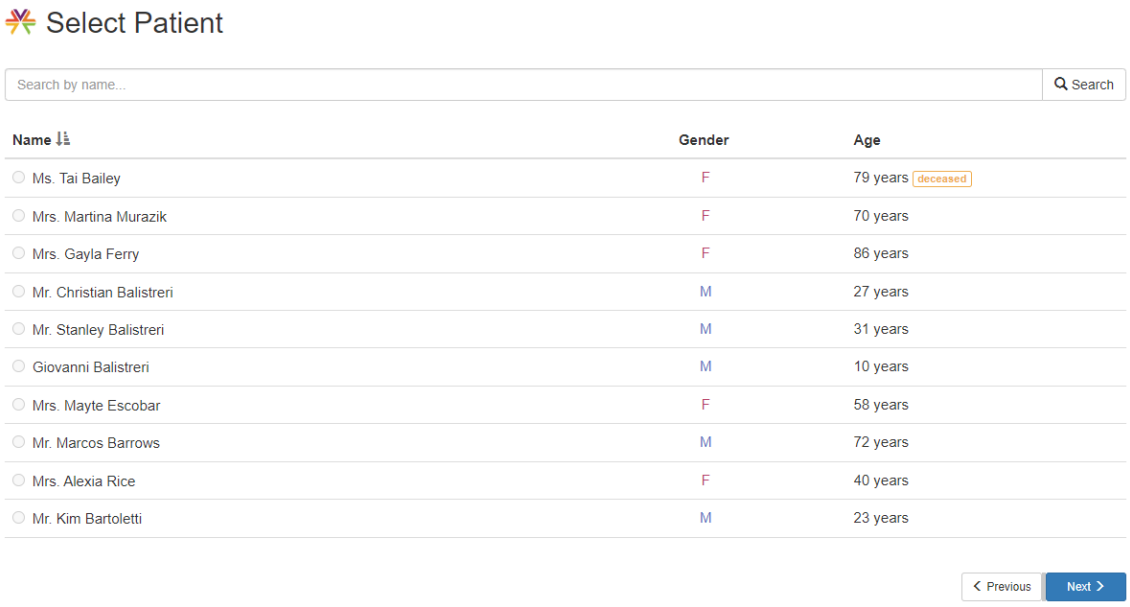
I detta demo väljer vi att klicka "Launch" och inte välja några alternativ. Man

skickas då iväg till en inloggningssida där en “practitioner” ska skriva in sina inloggningsuppgifter för att kunna logga in till journalsystemet. Denna practitioner kan exempelvis vara en läkare, eller en annan person som anses vara autentiserad för att kunna ha tillgång till journalsystemet. Eftersom detta är en sandboxmiljö, så funkar det med vilken practitioner och lösenord som helst. Men med ett riktigt journalsystem, så måste det vara en autentiserad användare.



Figur 5.2: SMART Login-sida

När man har loggat in och blivit accepterad som en autentiserad användare så får man välja en patient som skall skickas som en parameter till applikationen. De namn som kan ses i Figur 5.1 är exempelpatienter som SMART:s sandbox har skapat, men som ska simulera patienter i ett riktigt journalsystem. I detta exempel väljer vi patienten “Giovanni Balistreri”. Bredvid hans namn finns också ytterligare personlig info, nämligen kön och ålder.



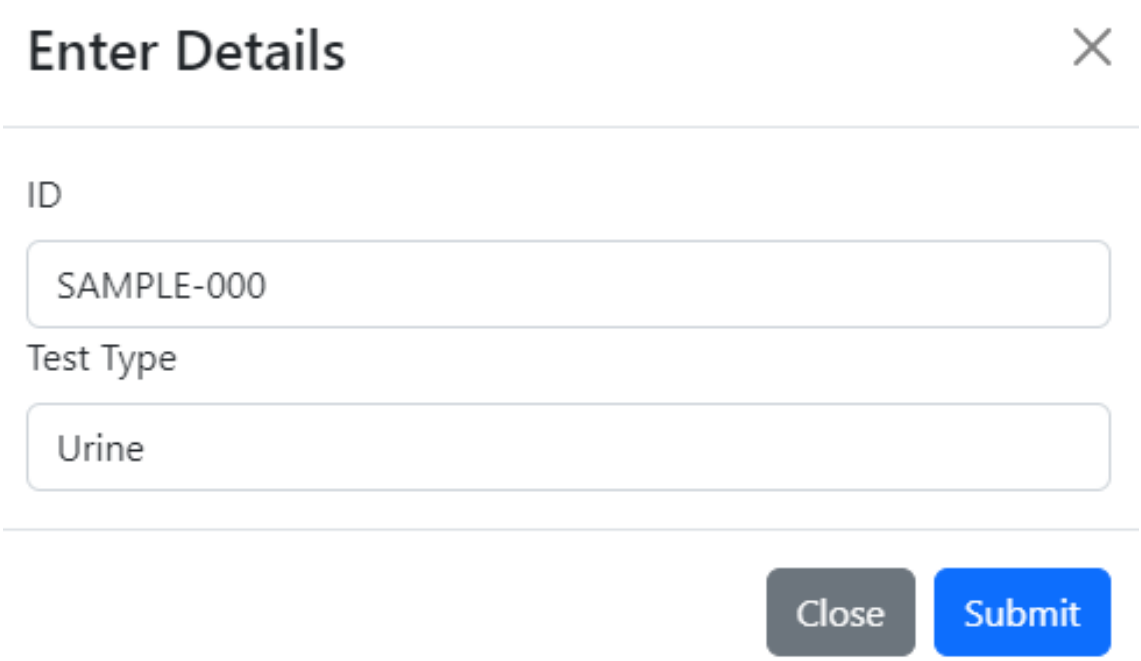
Figur 5.3: Sida där man väljer patienten

Det är förutsatt här att ingen har loggat in och valt Giovanni förut eller begärt uppgifter från hans profil i journalssystemet. När man har valt en patient så kommer huvudsidan för applikationen att visas. Som Figur 5.4 visar, så är utseendet på den färdiga applikationen likt den i Figur 4.3. I sidhuvudet finns Elyptas logga, och under den finns all information. När man endast har valt patienten så hämtar applikationen endast information från journalsystemet. Från Patient-resursen så vet vi att FHIR erbjuder mycket information man kan erhålla från en patient, men här visas bara den viktigaste personliga informationen om patienten i rutan "Patient Demographics". Under denna information finns en ruta där det står "Specimens" som syftar på patientens urin- eller blodprov. Eftersom inget urin- eller blodprov har gjorts på patienten så är denna ruta tom. Man kan däremot klicka på knappen "Request Specimen" för att skicka en förfrågan till labbet att ordna ett sådant.

Figur 5.4: Huvudsidan

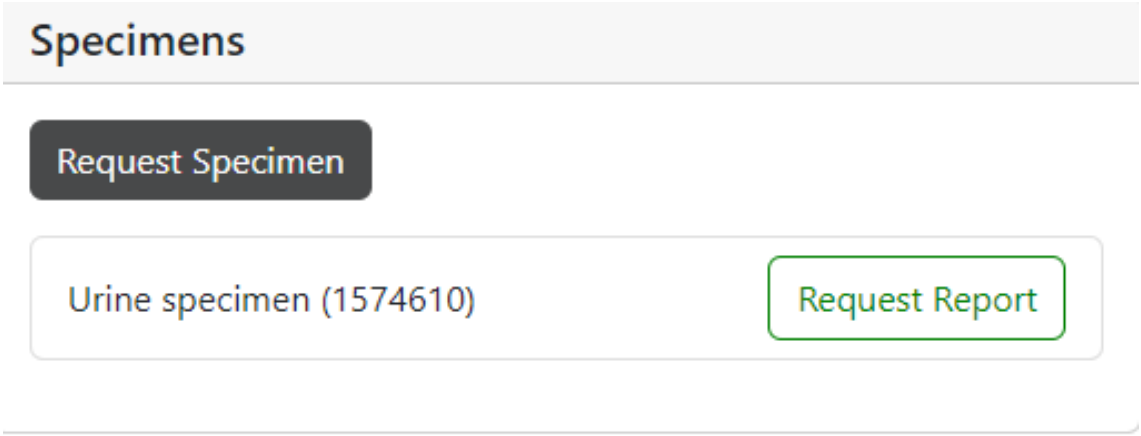
När man klickar på knappen så dyker en ruta upp på skärmen där man får skriva in ID-numret på testet man begär, såväl som vilket typ av test det är. Vanligen anger man "Urine" eller "Blood" i rutan där det står "Test Type". Själva ID-numret till

testet är egentligen unikt till patienten, men eftersom SKY-plattformen inte är igång för allmänt bruk så finns endast ett tiotal testprov att hämta vilket gör det svårare att visa i en demomiljö. Detta är något som kan implementeras i SKY senare. När man klickar på "Submit" så ska detaljerna skickas till labbet.



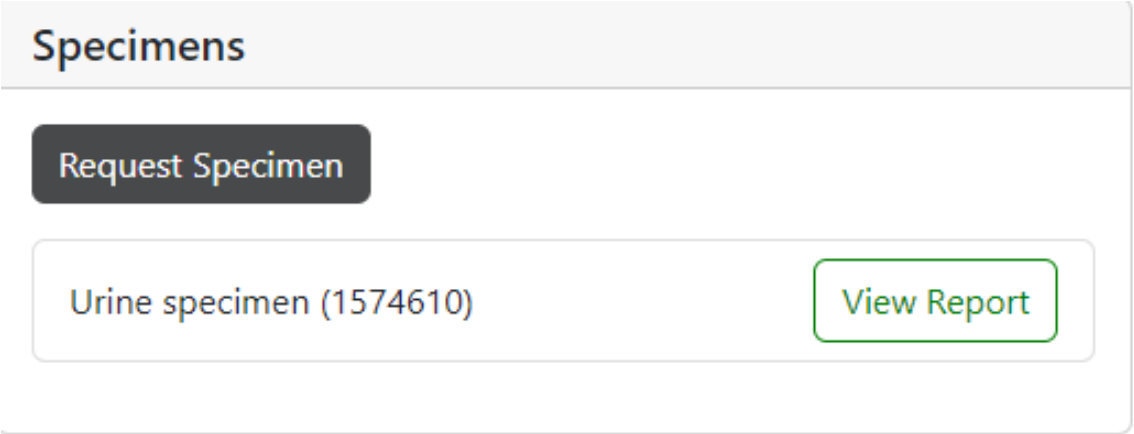
Figur 5.5: I denna ruta anges information om testet man begär från labbet.

När labbet har tagit emot detaljerna så dyker testet upp i listan. Labbet kan då klicka på "Request Report" som simulerar att labbet utför testet. Sedan skickas provet till Elypta som analyserar blodprovet.



Figur 5.6: Listan över urin- eller blodprov som skapats.

När man klickat på "Request Report" så simulerar det att Elypta skapar rapporten för att den skall visas i applikationen. Då uppdateras sidan med en ny knapp där det står "View Report".

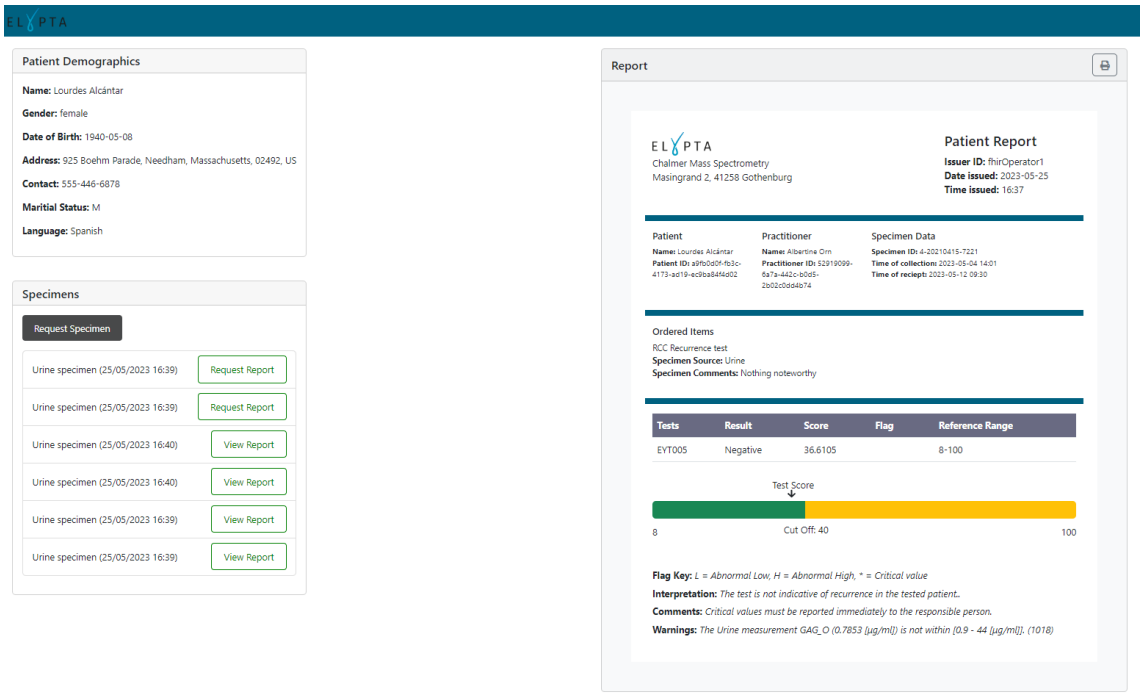


Figur 5.7: Knapp för att visa rapport.

Om användaren klickar på "View Report" så visas rapporten på sidan enligt Figur 5.8. Här finns information som nämnts tidigare, bland annat allmän information om patienten, användaren och själva provet. I rapporten visas även en visuell beskrivning på hur testets resultat ser ut. Detta görs med hjälp av komponenten "progress bar" som finns i React. Om "Test Score" befinner sig i det gröna området så är testresultatet negativt och patienten har med stor sannolikhet inte cancer. Om det istället befinner sig i det gula området så har patienten med stor sannolikhet cancer.

Figur 5.8: Den slutgiltiga rapporten.

Allt tillsammans ser applikationen i sin helhet ut som i i Figur 5.9. Användaren kan här fritt välja bland de olika tillgängliga rapporterna under runtime genom att trycka på “View Report”, samt antingen beställa nya prover genom “Request Specimen” eller nya rapporter på prover genom “Request Report”.



Figur 5.9: Applikationen i sin helhet.

5.2 Resultat från intervjustudie

Under vår intervjustudie kring hur mogen denna teknik är för att användas i svenska sjukhusmiljöer så har vi intervjuat Petter Wolff från Sahlgrenska Science Park. Petter Wolff är för nuvarande på uppdrag åt regeringen i den statliga utredningen "Utredningen om infrastruktur för hälsodata som nationellt intresse (S 2022:10)"[6]. Resultatet från intervjustudien baseras på svaren vi fått från honom.

I diskussionen med Petter kommer det fram att SMART on FHIR applikationer antingen används mycket begränsat eller inte alls inom svensk sjukvård. Millenium stöder SMART ON FHIR och kommer att försiktigt införa det i Västra Götalandsregionen. SMART finns även med som en komponent i många av regionernas strategi för IT-arkitekturen. Till exempel så delar Skåne och Stockholm plattformsarkitekturvision med Västra Götalandsregionen på denna nivå. Däremot saknas FHIR-API:er i produktion, vilket resulterar i en brist på sådana applikationer.

Vi diskuterar även om privata vårdaktörer aktivt arbetar med att utveckla SMART ON FHIR applikationer, där vi kommer fram till att det förmodligen handlar om enskilda personer som faktiskt arbetar på dessa applikationer. Detta på grund av att infrastrukturen för FHIR ännu inte finns, så värdet blir ofta för litet jämfört med att bygga sin applikation på andra ramverk och med andra metoder som en utvecklare redan kan.

Vi gick även igenom Petters tankar kring SMART ON FHIR applikationers potentiella framtida användande inom svensk sjukvård. Han berättade att om vi lyckas

etablera en likriktig nationellt standard avseende FHIR-standarden och enade om specifikationer inom denna, så finns det stora möjligheter till SMART att bli använt brett inom vården. En följdfråga till detta var inom vilka områden dessa applikationer skulle användas. Förmodligen kommer de byggas av specialiserade leverantörer som Elypta och av vården själv, som på ett portabelt sätt åstadkommer lösningar som fungerar i olika miljöer och kan integreras även i stora vårdgivares komplexa ekosystem.

Vi ville även ha Petters åsikt kopplat till ifall tekniken är mogen för att användas inom den svenska sjukvården ur ett utvecklingsperspektiv. Enligt honom är tekniken i sig mogen för det när en tillräcklig FHIR-infrastruktur har etablerats och kan använda SMART på ett produktivt sätt. Idag används e-SITHS för att autentisera användare och auktorisera applikationer i vården, så det är osäkert hur SMART:s 0Auth2-modell skulle samspela med detta. Detta skulle dock gå att lösa sålänge tid spenderas på att underöka noggrant hur man integrerar 0Auth2 med e-SITHS.

6

Diskussion

6.1 Utvärdering

Utvärderingen innefattar dels våra egna slutsatser som vi framställt baserat på de erfarenheter vi har fått från att ha utvecklat applikationen, och dels från den information vi har tillhandahållit från vår intervju.

Baserat på vår egna erfarenhet så kan vi sammanfattningsvis säga att det har som utvecklare varit tacksamt att jobba med en teknik som SMART ON FHIR. I början av utvecklingsskedet så spenderades mycket tid på att läsa om tekniken genom att söka upp dokumentation. Vi lade då märke till att det borde finnas mer dokumentation, och framförallt dokumentation som är mer lättläst. Det finns bra beskrivet vad alla resurser gör, men inte några exempel på hur de bäst kan användas i ett programmeringssammanhang. Vi vill också rikta kritik mot att det borde finnas mer dokumentation för andra programmeringsspråk än JavaScript. På SMART:s hemsida finns det förvisso referenser till guider kopplat till andra programeringsspråk, men dokumentationen är inte alls lika omfattande som för JavaScript. Det hade varit fördelaktikt för att bygga applikationer som inte ska köras i en webbläsare. Eftersom utvecklingsprocessen kan bli olika beroende på vilket journalsystem applikationen byggs för, hade det varit bra om SMART på något sätt kunde ha ett dokument eller liknande som beskriver vad som behövs för att applikationerna ska fungera på de mer populära journalsystemen. Till exempel hos Cerner som kräver client-id som parameter till autentiseringsfunktionen.

Förutom dessa oklarheter så var det en positiv upplevelse att bygga applikationen. Att hämta data från journalsystemet krävde väldigt lite kodning då "fhirclient" biblioteket skötte det direkt istället för utvecklaren. Om det uppstod frågor kring implementationer så krävdes det en enkel sökning bland dokumentationen för att få svar på det man letade efter.

Så utvärderingen från vår erfarenhet är att SMART ON FHIR är en tacksam teknik att jobba med vid utvecklande av denna typ av applikation. Det var både enkelt att arbeta med samt flexibelt nog för att utveckla många olika typer av applikationer, ett påstående som även stöttas av tidigare forskning kring SMART on FHIR applikationer [31]. Dock hade det kunnat finnas mer dokumentation gällande hur man får till en bra start på applikationsutvecklingen.

6.1.1 Etiska perspektiv

Säkerheten den naturligt mest relevanta etiska aspekten i utvecklingen av en SMART on FHIR applikation. Dock, som nämnt i projektets avgränsningar, har fokus i projektet inte legat på säkerhet och det har därav heller inte utvärderats djupare än konstateringen att det är viktigt att utreda om man skulle utveckla en applikation i produktionssyfte. Detta då säkerhet i SMART on FHIR applikationer är av yttersta vikt då känslig patientdata hanteras i kombination med att idag redan implementerade FHIR applikationer har visat på säkerhetsbrister[9].

6.1.2 Hållbar utveckling

Vi har utvärderat den hållbara utvecklingen utifrån de tre dimensionerna, det vill säga den ekologiska-, ekonomiska- och sociala dimensionen. Applikationen påverkar den ekologiska dimensionen positivt på så sätt att det bidrar till en pappersminskning, då journalerna är i elektroniskt format. Detta i sin tur minskar nedhuggning av skog och energin som går åt till att göra detta. Något som kan påverka denna dimension negativt är i sin tur energin som krävs för att upprätthålla en sådan applikation, men också avfall av äldre hårdvara. Påverkan vår applikationer hade haft på miljön om den använts brett är dock troligtvis minimal.

Ifall applikationen effektiviserar arbetet med journaler så gynnar även detta den ekonomiska dimensionen. Den främsta potentiella besparingen är en effektivisering av läkarens tid då mycket manuell handpåläggning elimineras genom en applikation, vilket hade frigjort tid för läkaren att fokusera på annat än administration.

Eftersom applikationens syfte är att i ett tidigt skede kunna upptäcka cancer så gör detta en positiv inverkan på den sociala dimensioner då man kan förebygga cancern tidigare, och på det sättet har man en större chans att rädda den människans liv. Det som har en negativ inverkan på den sociala dimensionen är till exempel ifall autentiseringen inte skulle fungera. Detta skulle leda till att oberöriga personer skulle få tillgång till material som de inte annars får ha tillgång till, något som strider mot den personliga integriteten och som kan skada samhället på längre sikt.

7
Slutsats

Under examensarbetets gång har en proof-of-concept applikation skapats med hjälp av verktyget SMART on FHIR. Applikationen uppfyller kraven satta av Gabriel från Elypta AB och är även godkänd av honom i efterhand. Alla funktioner och designval som planerats under arbetets gång är implementerade i applikationen.
För att kunna slutföra detta examensarbete har vi använt oss av de grunder och regler inom programmering som har lärts ut under relevanta kurser under programmets gång på Chalmers Tekniska Högskola. Efter arbetet har vi inhämtat ny erfarenhet och kunskap gällande tekniska verktyg och arbetssätt vilket ses som positivt med tanke på att vi efter detta kommer arbeta med utvecklande på heltid. Att ta till sig ny information och lära sig nya tekniker är en viktig del inom ingenjörsarbetet. Vi har även fått en helt ny erfarenhet hur det är att vara på ett företag och utveckla en produkt åt en extern kund.

Litteraturförteckning

[1] ”Teknikkonsulter med passion för hållbar produktutveckling”, *TogetherTech*, 2023, Tillgänglig: `https://www.togethertech.com/sv`, Hämtad: 2023-05-25.

[2] ”Advancing Cancer Detection”, *Elypta*, 2023, Tillgänglig: `https://www.elypta.com/`, Hämtad: 2023-05-25.

[3] “About Git“, *Git*, 2023. [Online] Tillgängligt: `https://git-scm.com/about`, Hämtad: 2023-05-24.

[4] “Who uses JIRA Software? “, *Atlassian*, 2023. [Online] Tillgängligt: `https://www.atlassian.com/software/jira/guides/getting-started/who-uses-jirafor-agile-teams`, Hämtad: 2023-05-23.

[5] “What Is FHIR? “, *The Office of the National Coordinator for Health Information Technology*, 2020. [Online] Tillgängligt: `https://www.healthit.gov/sites/default/files/2019-08/ONCFHIRFSWhatIsFHIR.pdf`, Hämtad: 2023-05-23.

[6] ”Utredningen om infrastruktur för hälsodata som nationellt intresse (S 2022:10)”, *Sveriges Riksdag*, 2023, Tillgänglig: `https://www.riksdagen.se/sv/dokument-lagar/dokument/kommitteberattelse/utredningen-om-infrastruktur-for-halsodata-somHAB2S10`, Hämtad: 2023-05-25.

[7] Joshua C Mandel *et al.*, “SMART on FHIR: a standards-based, interoperable apps platform for electronic health records”, *Journal of the American Medical Informatics Association*, vol. 23, no. 5, pp. 899-908, Sep. 2016. https://doi.org/10.1093/jamia/ocv189

[8] D. Bender and K. Sartipi, “HL7 FHIR: An Agile and RESTful approach to healthcare information exchange,” in *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, Porto, Portugal, 2013, pp. 326-331. https://doi.org/10.1109/CBMS.2013.6627810

[9] ”Critical flaws found in interoperability backbone: FHIR APIs vulnerable to abuse”, *SC Magazine*, 2021. [Online] Tillgängligt: `https://www.scmagazine.com/analysis/application-security/`, Hämtad: 2023-05-23.

[10] T. Benson and G. Grieve, “Implementing FHIR,” in *Principles of Health Interoperability (Health Information Technology Standards)*, pp. 397-416, Cham: Springer International Publishing, 2016.

[11] “SMART”, *SMART*, 2022. [Online] Tillgängligt: `https://smarthealthit.org/`, Hämtad: 2023-05-23.

[12] “Healthcare”, *Apple*, 2023. [Online] Tillgängligt: `https://www.apple.com/healthcare/health-records/`, Hämtad: 2023-06-19.

[13] “Azure SMART on FHIR”, *Microsoft*, 2023. [Online] Tillgängligt: https://learn.microsoft.com/en-us/azure/healthcare-apis/fhir/smart-on-fhir, Hämtad: 2023-06-19.

[14] “SMART on FHIR Tutorial”, Version 1.0, [Mjukvara], Cerner, 2023. Tillgängligt: https://engineering.cerner.com/smart-on-fhir-tutorial/, Hämtad: 2023-04-27.

[15] “Millenium Fakta om programmet“, *Västra Göralandregionen*, 2022. Tillgänlig https://www.vgregion.se/halsa-och-vard/vardgivarwebben/vardskiftet/millennium/fakta-om-programmet/, Hämtad: 2023-05-23.

[16] MDN contributors, “Learn to style HTML using CSS“, mdn web docs, [Online]. Feb. 2023. Tillgänglig: https://developer.mozilla.org/en-US/docs/Learn/CSS, Hämtad: 2023-04-27.

[17] React Bootstrap, Version 2.7.2, [Mjukvara], 2023. Tillgängligt: https://react-bootstrap.github.io/, Hämtad: 2023-05-23.

[18] W3Schools, “JSON - Introduction“, 2023. [Online]. Tillgänglig: https://www.w3schools.com/js/jsjsonintro.asp, Hämtad: 2023-04-27.

[19] W3Schools, “JSON Syntax“, 2023. [Online]. Tillgänglig: https://www.w3schools.com/js/jsjsonintro.asp, Hämtad: 2023-04-27.

[20] Hardik Shah, “React vs Vue – The CTOs guide to Choosing the Right Framework“, Simform, November, 11, 2022. [Online]. Tillgänglig: https://www.simform.com/blog/react-vs-vue/, Hämtad: 2023-04-20.

[21] Binmile, “React.js is Still Popular and Genuinely Dominates the Commercial Web Development Market‘´, LinkedIn, Mars, 16, 2022. [Online]. Tillgänglig: https://www.linkedin.com/pulse/reactjs-still-popular-genuinely-dominates-commercial-, Hämtad: 2023-04-20.

[22] Meta Open Source, “React The library for web and native user interfaces“, 2022. [Online]. Tillgänglig: https://react.dev/, Hämtad: 2023-04-20.

[23] ”What is a REST API?”, *Red Hat*, 2020, Tillgänglig: https://www.redhat.com/en/topics/api/what-is-a-rest-api, Hämtad: 2023-04-20.

[24] ”json-server”, *typicode*, 2023, Tillgänglig: https://github.com/typicode/json-server, Hämtad: 2023-04-20.

[25] Figma, “About Figma”, 2023.[Online]. Tillgänglig: https://www.figma.com/about/, Hämtad: 2023-06-20

[26] “The 2020 Scrum Guide”, *Scrum Guides*, 2020. [Online] Tillgängligt: https://scrumguides.org/scrum-guide.html, Hämtad: 2023-05-19.

[27] Tutorial - Building a JavaScript App, Version 1.0, [Mjukvara], SMART HEALTH IT, 2023. Tillgängligt: https://docs.smarthealthit.org/tutorials/javascript/, Hämtad: 2023-04-27.

[28] ”Resource Index”, *HL7 FHIR*, 2023, Tillgänglig: https://build.fhir.org/resourcelist.html, Hämtad: 2023-06-19.

[29] Figma, 2023, [Online], Kalifornien, USA: Figma, 2023. Tillgänglig: https://www.figma.com/, Hämtad: 2023-04-27.

[30] ”Petter Wolff”, *Sahlgrenska Science Park*, 2023, Tillgänglig: https://www.sahlgrenskasciencepark.se/contacts/petter-wolff, Hämtad: 2023-06-19.

[31] K. Wagholikar, J. Mandel, J. Klann, N. Wattanasin, M. Mendis, C. Chute, *et al.*, "SMART-on-FHIR implemented over i2b2," *Journal of the American Medical Informatics Association : JAMIA*, vol. 24, no. 2, pp. 398-402, 2017.

[32] Gustaver, M. (2020) A Chalmers University of Technology Master's thesis template for LaTeX. Unpublished.

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

GÖTEBORGS
UNIVERSITET

CHALMERS