CHALMERS

Development and implementation of RecipeKiosk system.

Commercial application using open-source and free components.

Master of Science Thesis in the Programme Software Engineering and Technology

Elvira Kim
Evgeniy Kim

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, August 2009

Development and implementation of RecipeKiosk system.
Commercial application using open-source and free components.

Elvira Kim
Evgeniy Kim

Examiner: Joachim von Hacht

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden September 2008

Abstract

This report contains description of the design and developemnt process of the RecipeKiosk software system and the choice of software components used for the system realization. The development of the system was initiated by request of JuzySystems AB.

The developed system is an interactive kiosk implemented using web-based client-server architecture and consists of hardware and software components.

The first part of the report is dedicated to the description of software analysis and design and explains requirements specification, use-cases, system architecture and user interface prototype.

For the implementation of system were used only free and open-source software components and the second part of this report describes the choice of components used. Among these components are Google Web Toolkit (GWT), Java Enterprise platform (JEE 5), GlassFish application server, PostgreSQL database management system and two distributives of Linux operating system – Puppy Linux and Ubuntu Server Edition

Foreword

This report describes results gained during the work on the thesis in Software Engineering at the Department of Computer Science and Engineering, Chalmers University of Technology. The thesis project has addressed needs of Juzy Systems AB. The project was realized under guidance and with great help of two supervisors: Joachim von Hacht from Computer Science and Engineering department at Chalmers University and Alexander G. Haaland from Juzy System AB. Joachim von Hacht has served as an examiner of thesis as well.

This thesis job has been carried out by two master students of Chalmers University of Technology: Elvira Kim and Evgeniy Kim.

Table of Contents

Introduction

Background

Interactive or electronic kiosk is a combination of hardware and software components used for providing some services such as access to information, making payments, etc. Usually it is an ordinary computer terminal with a display as output device and general input devices - mouse or trackball and keyboard. Also it can have some additional input/output means, such as printer, magnetic cards or bar codes reader, touchscreen and other. Terminal can be connected to a network for retrieving information or can work in off-line mode with all data stored locally.

Today electronic kiosks are used for many purposes. For example, it can be a self-service system in a library or an airport, terminals in big stores for checking products' prices and locations or informational kiosk in a museums, where visitors can get information about current exhibitions. There are many purposes and applications of kiosks, but several things are common for the most of them. First of all, kiosks are usually spread geographically – in different parts of one building or around the city or even country. Second, target users of kiosks are ordinary people probably with low level of computer skills or without any skills at all, that is why kiosks have to be easy for use, user-friendly and still be secure. They should limit an access to underlying system software and prevent improper use. And third, in the most cases information or service provided by kiosks is shared amongst many of them simultaneously and should be consistent and actual – they must have some means of communication between each other or with some central data storages.

While being well-known and widely used systems, in general electronic kiosks are still custom-made and complex software. Particular system supposed to be developed during this project is an ordinary specimen of electronic kiosks family. The project was initiated by JuzySystems AB and the idea of the system is to provide service for searching recipes and receiving news and information from stores. Kiosk are supposed to be used in a crowded public food products selling places, such as supermarkets. It should be easy to use, install and maintain.

What makes the development task more special is that the project has very limited budget. Therefore all software building blocks of the system should be free or open-source. It creates additional set of problems besides the actual system design and implementation. There is a problem of choice within numerous components and tools existing, but the chosen components should not be only applicable to the project itself – they must have licenses appropriate for using in commercial purposes.

Starting from this point and below the term 'kiosk' in content of this project will be used for referring to the particular system being developed during project.

'Terminal' will be used for referring to the hardware part of system, while 'software system' or 'application' will designate the software part.

Purpose

Main purpose of this thesis project is to develop a fully functional commercial system for searching recipes and accesing store information.

To accomplish this task the following steps will be done:

- analyze provided requirements;
- produce a high-level architecture of system and high-level choice of general components;
- investigate existing components and tools that can be used for implementation of each system's component and choose most appropriate from them;
- develop software systems using chosen components;
- evaluate achieved result.

The following restrictions are put on implementation of the system:

- only free and open-source software components and tools should be used.

The process of design and implemntation will be described in this report. Emphasis will be put on general and high-level issues and solutions. Also arguementation for the choice of software components is important. But there is no opportunity to describe the whole development process and details such as development methodology, all low-level design decisions or software testing methods.

Design

Design part of this report describes the analysis of requirements to the system, use-cases of system, software architecture and user interface.

Requirements

The kiosk system is a combination of specific software and hardware components and its purpose is to provide service for retrieving cooking recipes information from database, displaying recipes on screen and possibility to print them out. Supplementary system function is to display additional information, such as news about system, and play back an audio message with description of kiosk purpose.

There are two stakeholders for the project and the set of requirements was elicited during informal meetings between development team and Juzy Systems AB CEO. All requirements are divided into the three main categories: hardware constraints, functional requirements and other non-functional requirements. Each group is described below in more details.

Hardware requirements

- x86-based personal computer with touchscreen and without any other input devices in operational mode. Having only one input device - touch-screen - will simplify task of preventing user from unintended actions because user will not be able to use context menus or special key combinations. On the other hand this could make an interaction with the system easier for end-user – he/she will have all the controls directly on the screen and system developers will be forced to make the interaction logic simple and consistent. Computer must have 512-1024 MB of RAM memory, which should be enough for use by the most of modern operating systems and applications. For the permanent storage has to be used compact flash hard disk. Flash disks have big advantage compared to the traditional magnetic hard disk drives – they have no moving parts and noise. But the flash technolgy has disadvantage to deal with – expensiveness, that is why in case of the kiosk system hard disk has limited storage capacity of 1GB.
- Thermal printer for ability to print recipes and other information. Printer has to be compact, reliable, robust and easy for maintenance. It should allow to print recipes of any length and paper usage should be economical (possibly automatic paper cut after printing).

- Wireless modem and Ethernet network card can be used as an option for communication with outer world. The kiosk terminal can be placed in many spots with lack of necessary network infrastructure – cables, network switches, free telephone lines, etc., that is why mobile broadband connection is a supplementary option for the network communication. It can simplify installation of the kiosk. In case, if place of installation is not covered by mobile network, traditional wired connection can be used.

Functional requirements

- The user of developed system has two options for search for recipes: by recipe's title or by its categories, such as geographical origin, type of course or main ingredients. Both search options should be easy to understand and use – users should not be forced to execute complex sequences of actions to get result.
- When the search is done, the user gets resulting list of recipes and then he can choose any of the recipes from the list and browse it with more details on the screen.
- Recipes will contain following basic parts that should be displayed: recipe's title, ingredients, description and photo. During browsing through recipe user should have possibility to return back to search result.
- If the user wants he can print recipe using thermal printer connected to terminal. Only textual parts of recipe – title, ingredients and description – should be printed.
- When system is not used by anyone, recipes of the week should be displayed: one of three random or manually assigned recipes, which should remain the same during week. The user can choose any of week recipes for browsing with more details at any moment of time.
- Periodically news about place where kiosk situated, information about system updates, new functionality or any other kind of news will be posted and the user should have ability to browse this information.
- User interface should be appealing, consistent and simple for use, because the most of the kiosk users will be unexperienced computer users. The most of the actions should be possible to execute using 1-2 'touches'.
- The user should be prevented from accessing any kiosk functionality except one described in this specification, for examples users can't launch any other applications or have access to files on the terminal.
- The application will also use some audio messages to attract people to use it and help them to understand purpose of kiosk. This message should be played periodically when no one is using the system.
- Many terminals can be used simultaneously from different places and they should operate on the same recipes set.

- Statistical data of how many people have used the terminal and what recipes they have looked through and printed must be collected. To count the number of unique users the system should use timeout – if no actions were executed during some period of time since last action then system considers new action executed by new user.
- There should be additional utility for administration purposes – adding, editing and deleting recipes and news, viewing statistical data;
- The utility for administration use should be protected from unauthorized access by means of user-names and passwords.

Non-functional requrements

- The project has low budget and no additional cost should be paid for any used software;
- Because of teh low budget it also preferable that amount of the Internet traffic should be kept as small as possible;
- The kiosk system will be commercial, so the project is not open-source and the source code cannot be published;
- The kiosk system is the first project for the company, that is why there are no legacy systems or components that must be used and application can be developed using any programming languages, components and tools with regards to other requirements and constraints.

The customer didn't put strict time constraints for the project implementation, but roughly it was set to 6-8 month for the first working implementation.

Both the customer and the developers did not have any previous experience with a electronic kiosk developments and the requirements specification is produced in the general form with many necessary details left undiscovered. However during the development process some of the requirements were refined especially hardware configuration. And above is the final version of the customer requirements was given.

Use cases

After the requirements were defined and analyzed it is possible to produce a more formal specification. This specification is made in form of use-cases, which describe who will use the system and how they will do it.

There will be two actors who will use the system.

- **User** is represented by any person, who employ system services through the publicly available kiosk terminals.
- **Administrator** is an authorized person, who can control and manage the system – add or edit recipes, check statistics, etc. He/she can do it using any available computer.

In the current chapter the term 'database' is used. In the context of use-cases database is a general abstract place where all persistent information is stored (recipes, news, statistics journal, etc.) and concrete implementation of such functionality is not significant at the moment.

Also in the use-case description the term 'statistics journal' is used. It means a some subsystem of the kiosk application, which will allow to store and keep different events occurred during system operation. The event will include at least time and terminal of occurrence and type. This journal will be stored permanently in the database.

User's use-cases

All the following use-cases are available for unauthorized access from any kiosk terminal.

1.1 **User registration.** For possibility to calculate amount of users utilized kiosk, every time new user touches screen "new user" event should be stored in statistics journal. To distinguish users timeout mechanism will be employed: system will have two states: active, when someone uses it, and inactive, when timeout period has passed since last user activity. Thus "new user" event is switch from inactive state to active.

1.2 **Search recipe by title.** User chooses option to search recipes by title, then he types one or several words, which are used to search through recipes' titles. Words are matched against recipe's title and in case if all entered words have been matched recipe will be added to search result. After search is done result is shown to user.

1.3 **Search recipe by categories.** User chooses option to search recipes. Then he chooses set of recipe's categories. Example of categories are recipe's main ingredient, geographical origin, etc. If recipe belongs to all chosen categories it will be added to search result. After search is done result is shown to user.

1.4 **Browse search result.** After one of search variants was chosen and performed and result recipes collection is shown to user. Recipes should be shown in short form for the sake of fitting more recipes on screen. User interface will have usable means of navigating through resulting collection in case of it contains too many recipes and cannot fit on screen. During browsing through resulting collection user can choose any recipe to be shown in full form.

1.5 **Choose and browse recipe.** When user is browsing recipes retrieved after search, he can choose any recipe for full view. Full view includes detailed ingredients and cooking description with full-size recipe photo. Every time user chooses any of recipe to browse corresponding event should be added to statistics journal. This event should also include what recipe was browsed.

1.6 **Choose and browse week recipe.** When user starts using kiosk he will see one of three week recipes assigned manually or randomly for current week. Recipe will be shown in full view (see Use-case 1.5) and is chosen automatically from current three week recipes, but user will have possibility to choose any of week recipes to be shown in full view. Browsing of week recipes should be added to statistical journal in a way similar to described in Use-case 1.5.

1.7 **Print recipe.** During browsing chosen recipe in full view (both after search or one of week recipes) user can print it out using thermal printer connected to terminal. Information to be printed is full recipe description except recipe photo. Printing of recipe should be recorded in statistics journal and record should contain what recipe was printed.

1.8 **Browse news.** At any moment of time user can browse information about system updates, place where terminal located, etc. Each piece of such information, news for simplicity, will have along with content start and end dates to be displayed. Also news will be assigned to one or more terminals, on which they should be shown. Every time when user chooses news to be displayed corresponding event has to be written to statistics journal.

Administrator's use-cases

As it was stated above administrator can use any ordinary computer for accessing administration functionality, but this functionality should be protected from unauthorized access.

2.1. **Login to administration service.** Every time administrator wants to use service he needs to provide authentication information in form of user name and password. But he should not login for every action he wants to do. Once logged in his session should last until administrator logs out or he was inactive for defined amount of time

2.2. **Browse list of recipes.** Administrator can get list of all recipes stored in database with possibility to filter out list by title or description content and categories. Recipes list will contain only recipe title to be able to show more recipes on screen.

2.3. **Add recipe to database.** Administrator can add new recipes to database at any moment. For adding a new recipe he has to enter its title, description, choose recipe's photo and select appropriate categories.

2.4. **Edit recipe in database.** During browsing recipes list (Use-case 2.2) administrator can choose any recipe for editing it. When he do it recipe will be shown in full view – title, description, categories and photo, with possibility to edit update any of these components. When updates are made administrator can save them in database.

2.5. **Remove recipe from database.** When administrator look through recipes list (Use-case 2.2) he can choose any of them for deletion. But before recipe will be deleted from database administrator must confirm it once more. After confirmation recipe will be erased from database permanently.

2.6. **Browse list of news.** Administrator can get list of all news stored currently in database. News list will contain news title and description, time period, when news will be displayed, and terminals, where news will be shown.

2.7. **Add news to database.** Administrator can add news to database at any moment. For adding a news he has to enter its title, description, choose period, when it should be displayed, and select appropriate terminals, where news will be shown.

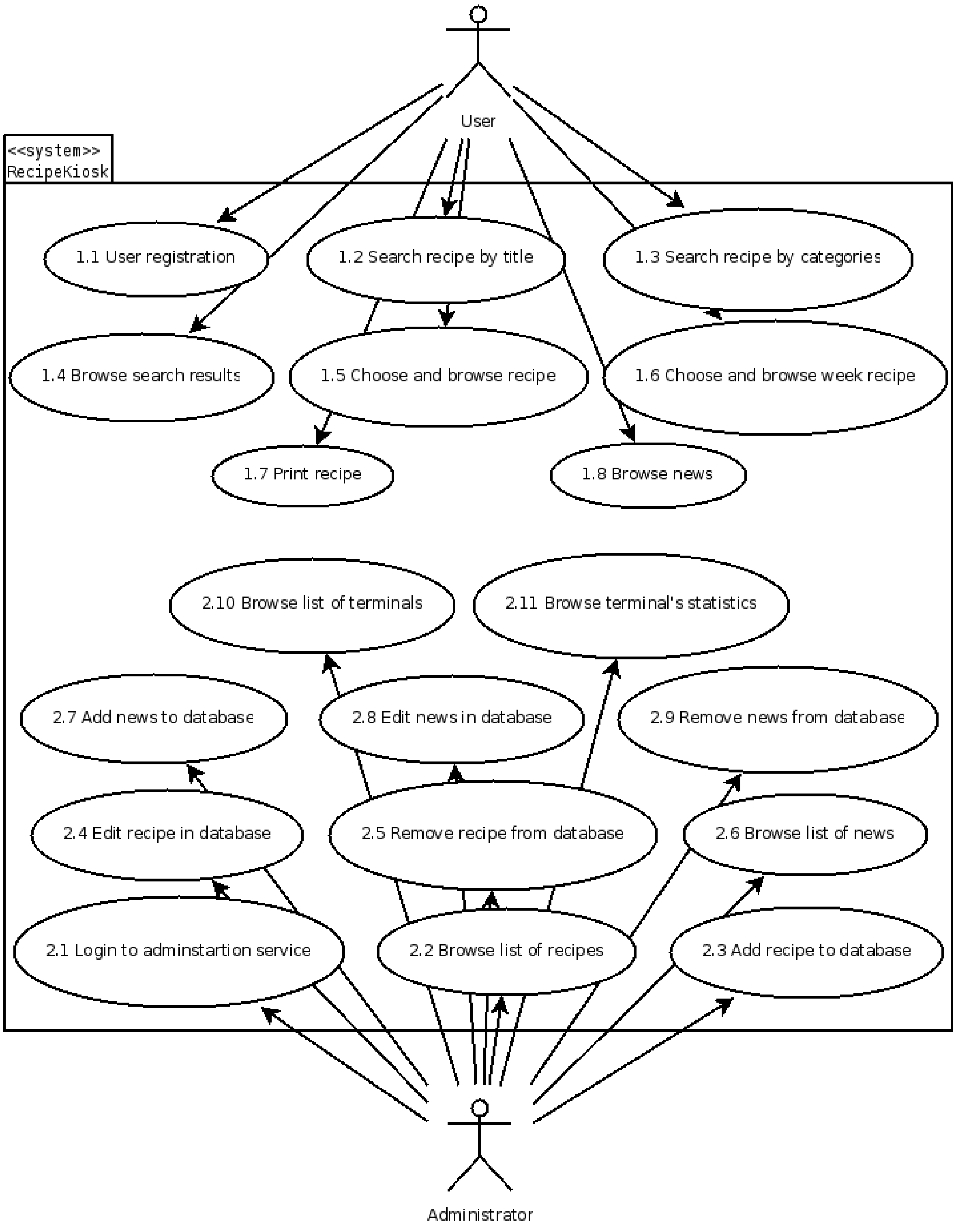
2.8. **Edit news in database.** During browsing news list (Use-case 2.6) administrator can choose any news and edit it. When he do it all parts of news will be shown – title, description, terminals and time period, with possibility to edit update any of these components. When updates are made administrator can save them in database.

2.9. **Remove news from database.** When administrator look through news list (Use-case 2.6) he can choose any of them for deletion. But before news will be deleted from database administrator must confirm it once more. After confirmation news will be erased from database permanently.

2.10. **Browse list of terminals.** Administrator can browse list of terminals registered to the system. For each terminal there should be information if terminal is working currently or not.

2.11. **Browse terminal's statistics.** While browsing list of terminal administrator can choose any terminal to browse detailed information about. This information can include place where terminal is located and statistical data of terminal's use. Administrator can choose period for which statistics should be calculated. Statistical data will include how many people used terminal, what recipes where browsed and what recipes were printed.

Following is use-case diagram drawn from use-cases described above:



User
<<system>>
RecipeKiosk
1.1 User registration
1.2 Search recipe by title
1.3 Search recipe by categories
1.4 Browse search results
1.5 Choose and browse recipe
1.6 Choose and browse week recipe
1.7 Print recipe
1.8 Browse news
2.10 Browse list of terminals
2.11 Browse terminal's statistics
2.7 Add news to database
2.8 Edit news in database
2.9 Remove news from database
2.4 Edit recipe in database
2.5 Remove recipe from database
2.6 Browse list of news
2.1 Login to adminstartion service
2.2 Browse list of recipes
2.3 Add recipe to database
Administrator

Image 1: Use-case diagram

System architecture and design

System under development is not trivial and cannot be implemented directly after requirements elicitation. Preliminary design phase is required to make development process controllable and to achieve desired result.

After design phase a system architecture can be produced. To describe the system architecture in more consistent way several representations from different points of view are used. These representations are also called views.

Description of developed system's architecture contains following views:

- Logical view. This view is a description of high-level components responsible for required functionality and associations between these components.
- Deployment view. This view shows disposition of hardware components and mapping of software components onto them.
- Data view. This view describes structure of data entities that will be used by system and stored in database.

Logical view

Logical view on the system architecture focuses on required functionality.

This view is logical decomposition of system into components with regards of functional requirements which will be satisfied by every component. Also relationship between these components are shown.

Following diagram represents logical view on architecture:

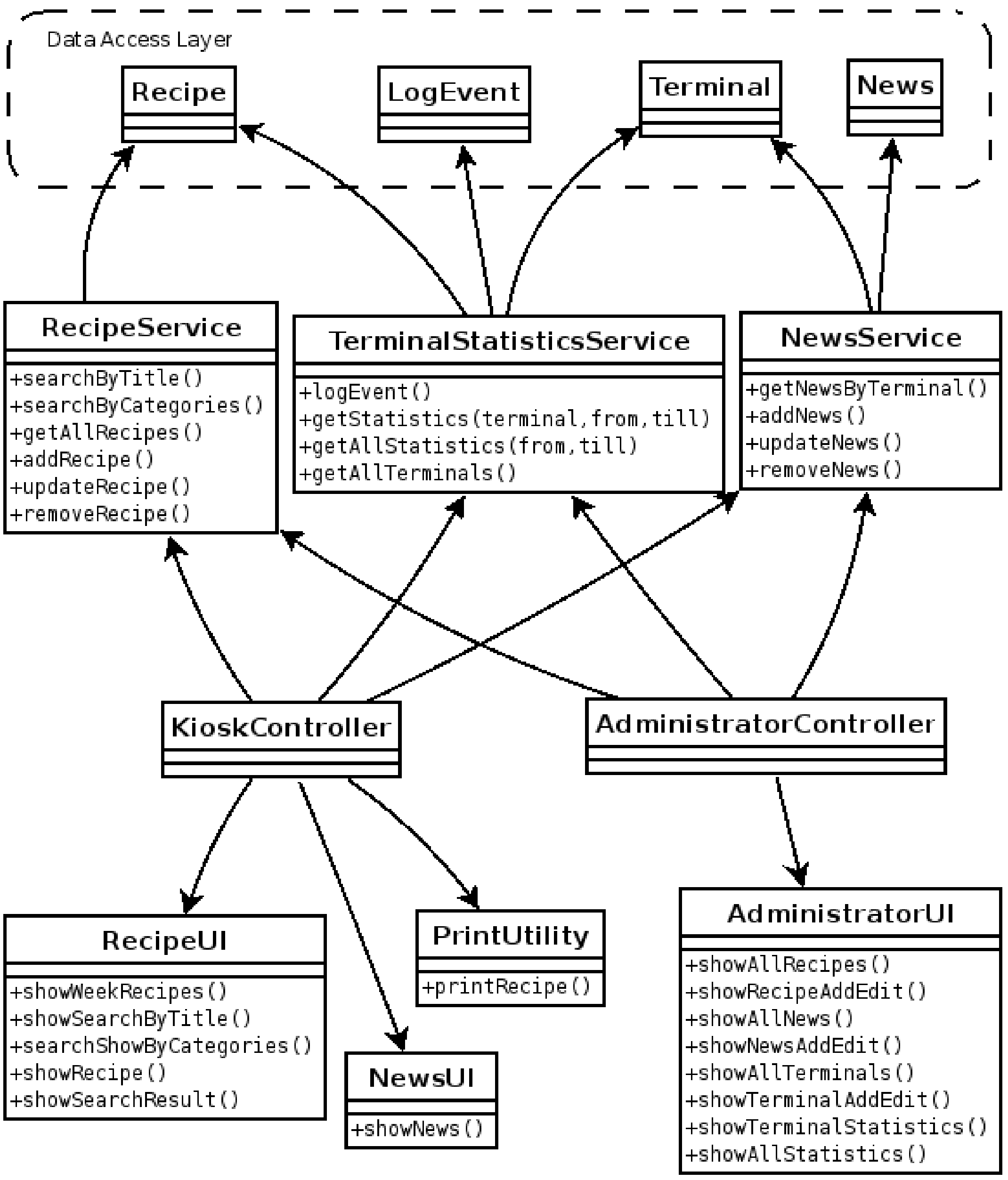


Image 2: Logical view

Structure above closely resembles model-view-controller (MVC) architectural pattern.

Model part of pattern contains three components: RecipeService, TerminalStatisticsService and NewsService. This part is responsible for management data retrieval and modification activities.

View part is represented by RecipeUI, NewsUI, AdministratorUI and PrintUtility. These blocks provide data presentation functionality both on screen and on printer. RecipeUI renders interfaces for recipe search, search results and recipes themselves, but it is not providing means for recipe management: addition, update or deletion. Recipe management interfaces are included in AdministratorUI. Also AdministratorUI is responsible for news management user interface and displaying statistical data. NewsUI is responsible for displaying news and navigating through them.

Two intermediate components – KioskController and AdministratorController – manage data flows between model and view parts. They are supposed to handle user inputs and send appropriate message to model. After responses from model received controllers update view depending on information contained in them. Additional important task of AdministratorController is execute authentication and authorization functionality before administrator can use rest of its services.

Recipe, LogEvent, Terminal and News components on the top of diagram correspond to data entities used for informational exchange between view, controller and model and they are depicted with more details later on in data view.

Note, despite of fact that class diagram is used for representing logical view in reality components shown on it can be implemented using several programming language classes. Also for the sake of keeping diagram simple only large significant components are depicted.

Deployment view

Having demand that all terminals will share same recipe data most obvious deployment architecture is client-server architecture. Client-side part of system, or client for shortening, is point of access to the system by users and administrator. Server-side part, or just server, must keep shared information in consistent way and provide access to it by clients.

To store and record recipes, news and statistics information relational database system has to be used and server-side will at least include database system.

Client-server architecture can be realized in many different ways. One of important issues is how much functionality client will have and which way it will be implemented. There are many ways of how this issue can be solved. It could be full-sized desktop application (thick client) that connects directly to database for retrieving raw information and then rest of data processing is done on client – data transformation, user interface rendering, etc. Serious disadvantage of this approach is updates of client-side application part. If system patches will be issued or new functionality will be added to application, then new version of client side should be installed on every terminal. It has be done in automatic mode as terminals will be located in different places and with thick client approach it would be harder to implement these unattended updates.

Second option is to have most of the functionality stored or executed on server-side. Client-side in this case will be responsible only for rendering user interface, capturing input message and communicating with server – thin client. This approach is usually based on using web browser as container for thin client. Thin client architecture allows to avoid software installation and all updates will automatically be applied on every client with reloading browser's content. Having this advantage web-based thin clients can at the same time bring desktop application alike user interface experience and have quite complex calculations and data processing being executed on client-side. Additional big advantage of web-based architecture is cross-platform portability. Most of web browsers have been ported on many platforms, they are supposed to show similar result after processing responses from web server and they have additional plug-ins for handling different type of media – video, audio, flash. As disadvantages of web-based approach some issues can be mentioned, such as non-trivial ways of preventing users from access to operating system tool and utilities or implementing low-level client functionality, such as printing.

In case of developed system web-based architecture is chosen. Reasons behind this choice are:

- ease of system updates;
- simplification of user interface design and modification;
- flexibility based on rich choice of available components, that can be used for realization;
- platform-independence based on fact, that most of available components

either have versions for many different platforms or have analogues on different platforms.

Physical architecture

Simple diagram for physical view of system architecture looks following way:

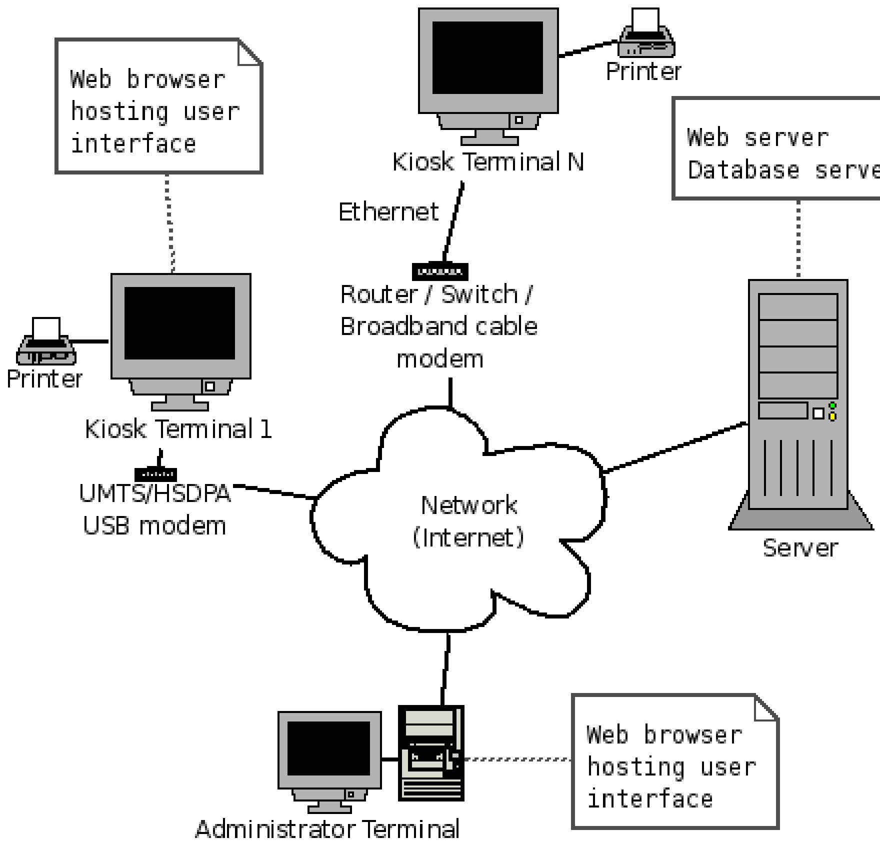

Web browser hosting user interface
Printer
Kiosk Terminal N
Ethernet
Web server Database server
Router / Switch / Broadband cable modem
Printer
Kiosk Terminal 1
UMTS/HSDPA USB modem
Network (Internet)
Server
Web browser hosting user interface
Administrator Terminal

Image 3: Web-based system architecture (hardware components)

Scheme above is classical client-server architecture. Client is a personal computer equipped with touch screen as it stated in requirements. Besides all hardware components of terminal (except printer and modem) are contained inside touch screen's body (all-in-one computer), which makes client more protected and appealing and eases hardware deployment.

Client has a thermal printer connected to it. Thermal printing is chosen for Thermal printer is chosen due its faster and quieter printing comparing to dot-matrix, laser and ink printers. Thermal printing is also more economical than other technologies since only paper is consumed. Another advantage is rapid and ease paper refill. And though thermal paper is more expensive this printing is most attractive for use on developed kiosk system.

For connection to server client terminal can use two options: Ethernet network card or wireless broadband modem. Having two options provides flexibility to developed system and allows to choose most appropriate solution for particular place of kiosk installation.

First connection option assumes there exists device which provides access to public network. Terminal is connected to this device using Ethernet local network and it can be cable modem, switch or router used as this device.

For second option is chosen mobile broadband modem. Modem uses mobile network as communication medium and works similar to mobile phones. To have appropriate data exchange rate both modem and mobile network should support HSDPA (High-Speed Downlink Packet Access) protocol. HSDPA allows to have high data transfer rate more than 1 Mbit/sec, which is comparable with traditional cable technologies. Modem is connected to terminal using USB port.

In both cases connection should have at least 1 Mbit/sec bandwidth for comfortable kiosk usage. This concerned with big amount of images for recipes to be fetched from server.

Server side differs significantly from client side. To be able to service many clients simultaneously server has to have more computing power, memory and better network connection. Additional important issue is reliability of server hardware components – server failure will lead to all kiosk stop function. As solution for server-side hardware configuration and network connection it is decided to use external service of server hosting. This solution can significantly decrease amount of resources required on initial stage of project and provides required level of reliability. Moreover instead of having dedicated hardware server virtual server hosting is chosen. Having virtual server allows to have less powerful server on initial phase of testing with few clients and then to scale server performance with less expenses on later stages, when number of client will grow.

Administrator's terminal can be any personal computer connected to Internet not necessarily dedicated for developed system administration. It has no specific hardware requirements and has only to be able run operating system with graphical user interface and web browser. There are also no specific requirements on network connection means and bandwidth, only measure is personal perception of comfortable application use.

Because end-point devices are spread geographically all data communication between server, kiosk terminals and administrator computer is supposed to run over Internet and no additional network infrastructure or technologies are required (dedicated lines, virtual private networks etc.).

Functional blocks of developed software system described in logical view of architecture can be distributed different ways along hardware components of web-based system. Some functionality, by intuition, supposed to be placed on the server-side, for example, data access layer. But most components and functionality can be placed at both server- and client-side or even one part of logical component can run on server while second be hosted on client.

Software-to-hardware components mapping

Following diagram depicts mapping between software and hardware components of the system:

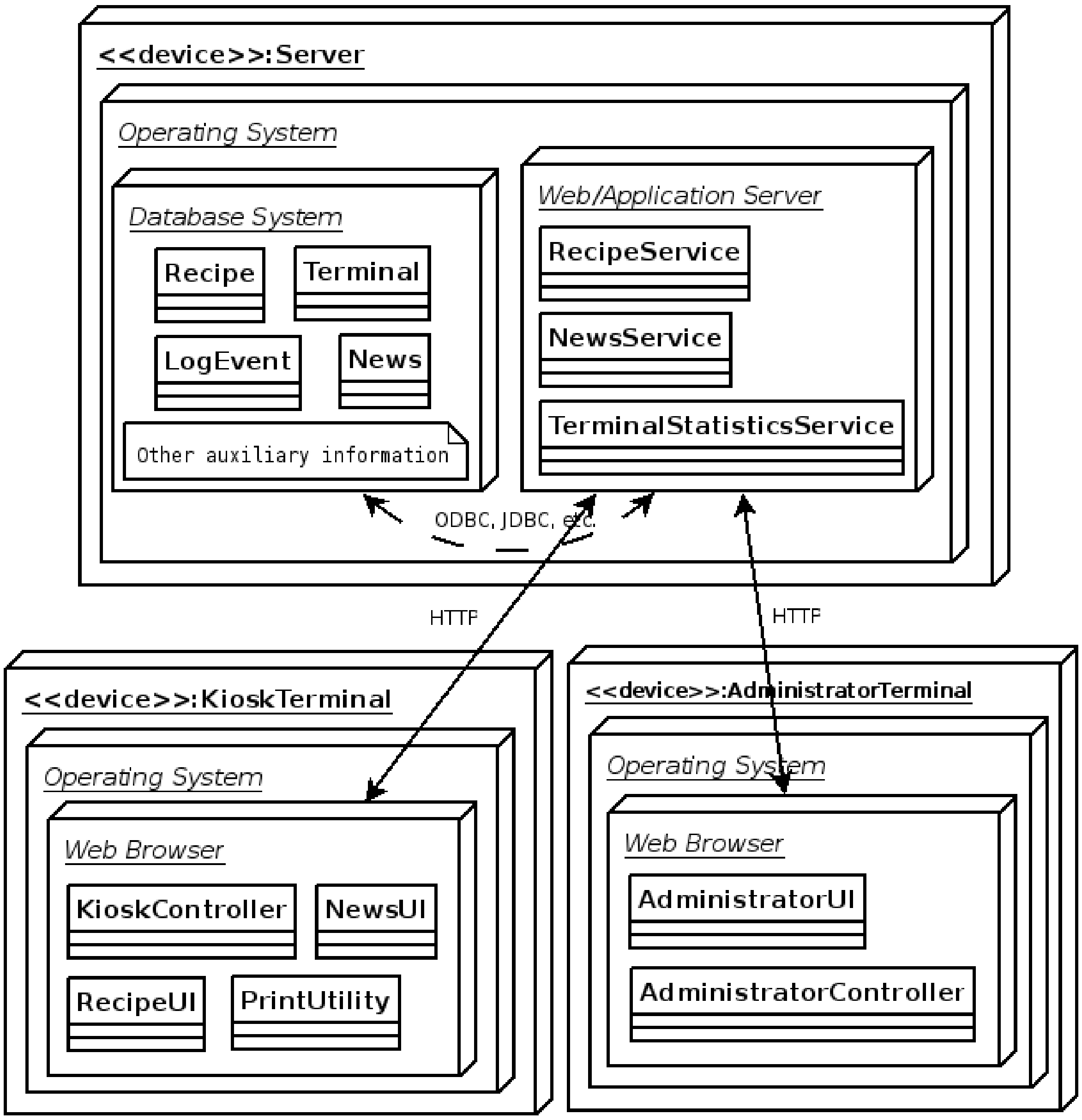


Image 4: Deployment diagram

Three main hardware parts are server, kiosk terminal and administrator's computer. All three of them will host some operating. Administrator and kiosk terminals must have an operating systems with graphical user interface as all user interaction with system will be done through application interface in web browser, which will have many images and graphical control elements (buttons etc.).

Server does not require graphical interface, it is even more preferable not to have one on server, because graphical interface will consume processor time and both operational and permanent memory. Since server is virtual machine hosted on external side without direct access to it all server operating system administration will be carried out in command-line terminal remotely through SSH (secure shell) over VPN (virtual private network) connection to hosting company's network.

Two other software components required on server are database server software and web or application server. They both will be run on the same machine to simplify management and due economical reason, as only one virtual server will be hosted initially. In case of need they can be separated later or even several instances of application server can be run for load balancing.

Database server will keep all permanent information supposed to be shared among kiosk terminals and other auxiliary data. Besides textual information, numbers and dates database will store recipes photos. Though many web applications keep images as separate files on the file-system, because it simplifies image updates and access, in case of developed system it will be rare case when recipe photo should be changed and having recipes photos stored in database will ease creating data backups and restoring data in case of failure.

Application or web server software component is required because client part will be run in web browser and can't access database directly, in contrast to desktop applications. Three logical components will be executed inside application server – RecipeService, NewsService and TerminalStatisticsService, responsible for retrieval and management data in database. Request to these service and corresponding replies will be carried out over HTTP connection Application server will process HTTP request arrived from client web browser and pass control to appropriate logical component. Then generated responses will be transmitted back to clients browser. Additional function of application server is to keep shared copy of client part of system. This copy will be transferred on kiosk terminal after initial request from client web browser. It is discussed in more details below. when client part is described, and later in chapter about user interface.

Configurations of kiosk terminals and administrator's point of management are similar in terms of software components required. As it said previously both will have operating systems run in graphical mode and also both requires web browser for executing client part of the system. Only difference is in restrictions applied on choice of operating system and browser. While kiosk terminals all will have same operating system and web browser for simplification of their installation and configuration and choice should be made in the beginning of system implementation. Especially web browser should be defined in advance because kiosk user interface supposed be quite specific and it will be harder to maintain its cross-browser compatibility, which is not necessary as development team has control over software configuration in contrast to common web application environments, when client part can have different web browsers and operating systems. Administrator's terminal will be standard personal computer and user interface of administrator's client part is supposed to be more standard and simpler, that is why it is not required to define exact operating system and web browser in advance.

As it was said kiosk and administrators user interfaces will be hosted inside web browser. Traditional dynamic web applications use HTML pages for providing user interface and after each request from web browser server generates and sends page in reply. This leads to page reload and round-trip delays visible to user. Acceptable for most web-applications in case of developed kiosk terminal it can distract users from using service.

There are number of technologies existing for providing desktop-alike behavior web applications. Some of them require additional plug-ins or add-ons to web browser (Microsoft Silverlight, Adobe Flash), while other can be employed directly in browser (Ajax). Common bottom of all of them is that shared code for user interface is stored on server and upon request transmitted to client side and then is executed there.

To provide richer user interface and seamless navigation across application second approach is chosen. It means that NewsUI, RecipeUI, AdministratorUI and PrintUtility logical components will run on client side inside web browser. Along with these components also controller part of described MVC architecture described above will be executed on client side. Additional advantage of having user interface functionality running on client is possibility to decrease amount of data transferred between server and client.

As it was said above all code of RecipeService, component for recipe data retrieval and management, will run inside application server. Searching through set of all recipes is supposed to be done also on server side. But decision is made to provide additional alternative approach, when part of this service's code will run in web browser along with user interface and controller. Specifically search methods can be transferred to client side as it shown on image below:

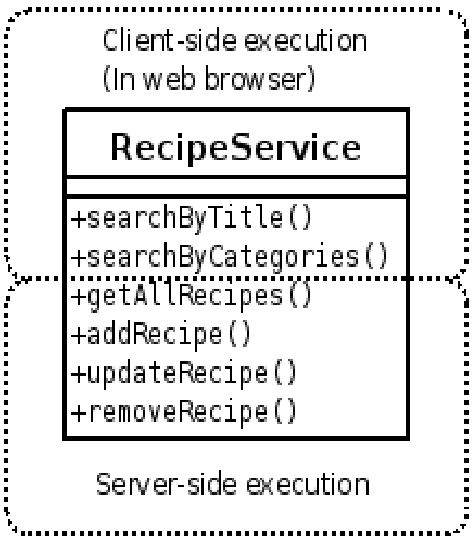


Image 5: Business logic separation

This approach has some advantages comparing to initial one. It assumes all recipes, except their images, will be fetched once to client on application start. Then different kind of search will be done on local copy of recipes. It can significantly improve application responsiveness and decrease time of search by eliminating data transfer delays upon each search. To have updated copy of recipes application can be reloaded after given period of time, for example once in a day, or new copy requested without reload or eve kiosk restarted.

Additional advantage of such approach is possibility to decrease amount of traffic between server and clients. In case when several users will receive significant amounts of recipes as search result sum of traffic can exceed total size of all recipes.

Only issue for this approach is total size of all recipes. This will be calculated during testing phase when appropriate amount of recipes will be added to database. In case if size of recipes' data will be too large for keeping them all in primary storage of kiosk terminal server-side search can be employed.

Data view

Data view represents architecture in terms of data used for informational exchange between system components and which stored in database. Though logical view already contains high-level data model, data view describes it in more detailed way.

Below is shown ER diagram for data, which will be used in system:

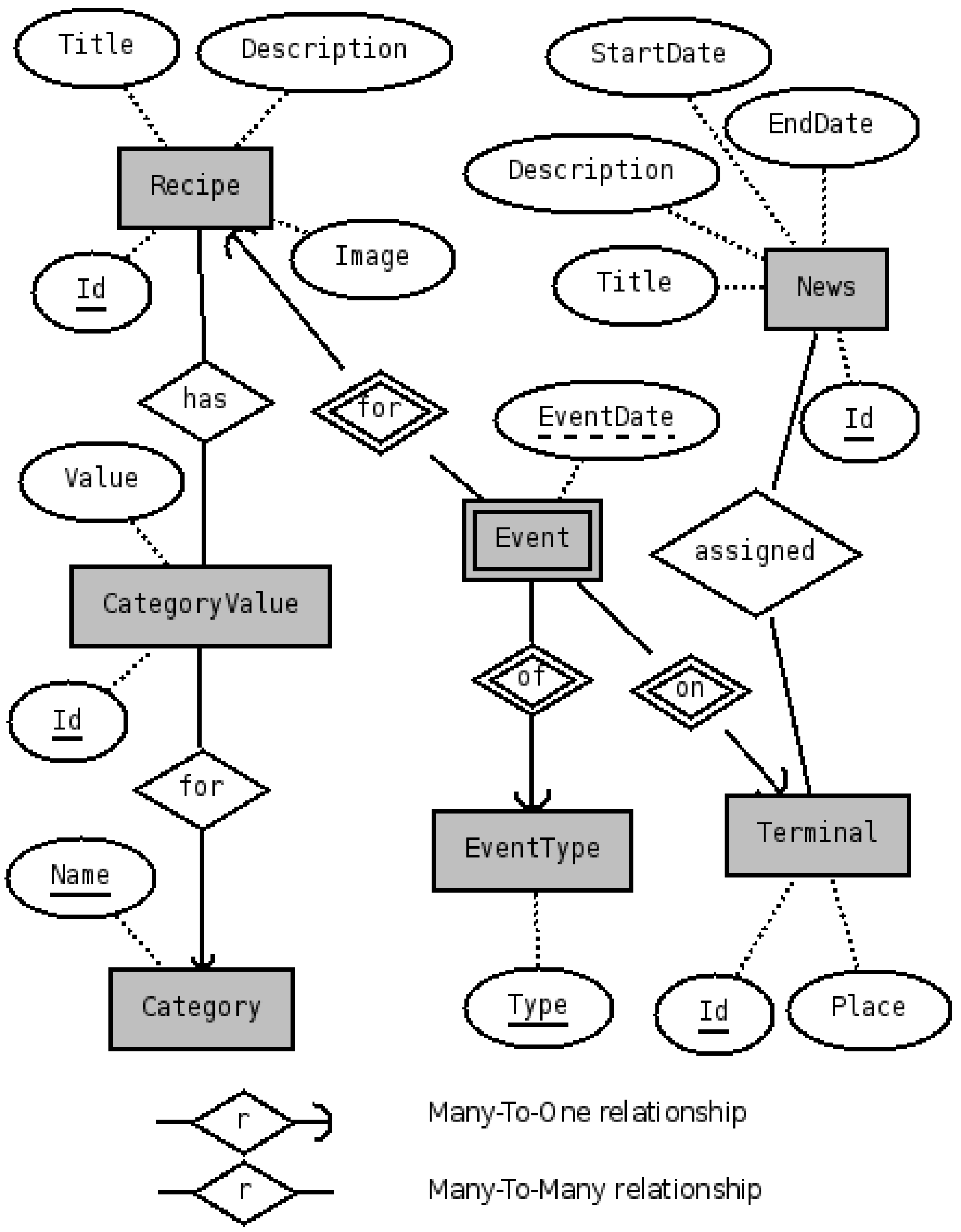


Image 6: ER diagram for system database

Rectangular boxes on above diagram designate main data entities representing concepts used by kiosk application. These entities can correspond to tables in relational database and their meaning is straightforward. Worth to mention are Category and CategoryValue entities. First one is used to depict groups of specific recipe attributes, such as recipe origin, main ingredient, type of course, etc. While second represents concrete values inside these groups, for example Sweden, China for origin category, main course or desert for type of course category.

Box with double border, - Event - is a weak entity, which means its primary key depends on primary keys of other entities it has relationships with.

Ovals connected to entities are attributes and they corresponds to data fields in relational table. Attributes with underscored (both solid and dashed lines) labels are primary keys or components of primary keys.

Rhombuses depict relationships between entities. Many-to-many relationships having arrow-less lines on both sides need to have separate table in terms of relational databases. While many-to-one relationships can be implemented by having primary key of 'one' side be included in table created for 'many' side.

Kiosk user interface prototype

Following are blueprints of user interface for different states of the system and they do not reflect many graphical details (images, font styles, colors) of final user interface. As it was said above, developed web system will not use traditional approach, when whole page is reloaded, and it means that prototypes below are not designs of separate web pages, but rather screen-shots of user interface during system usage and only some subset of interface elements are updated, added or removed, while whole interface stays consistent and user gets seamless experience of interface functioning.

Some elements of user interface are common and should be displayed regardless of current state of application and only size, color or content of these elements will change. One of such elements is tab bar on the bottom with two tab buttons – 'Recipes' and 'News'. User always see and use them only size tab button and size of its title will change depending on which tab is active. Second element is label on the top of screen which change content according to what part of user interface is active now.

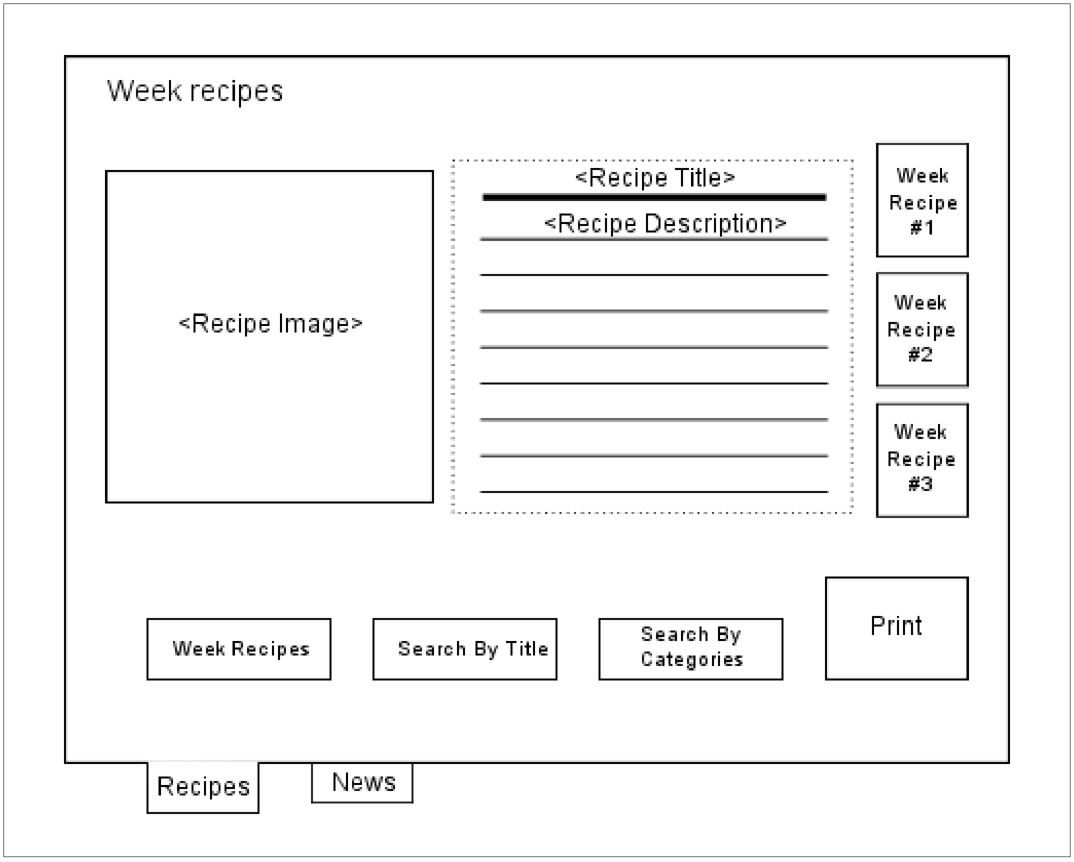


Image 7: Week recipes screen

First blueprint shown on Image 7 is initial page, which user sees when he starts using kiosk.

User can see one of randomly chosen week recipe in full mode with full-size photo and description. If recipe's description can't fit allocated area user can scroll it down and then up again. Also user can choose another week recipe using buttons on the left side of screen. This buttons are represented by thumbnail picture of recipe's photo and its title.

Three buttons on the bottom used to switch between week recipes browsing and two search options. When week recipes are browsed corresponding button is marked by, for example, different color and/or font style and label on the top of the screen displays appropriate information.

Additionally 'Print' button is displayed in right bottom corner, pressing on which user can send currently displayed week recipe for printing.

When user press 'Search By Title' button he will be shown screen on Image 8.

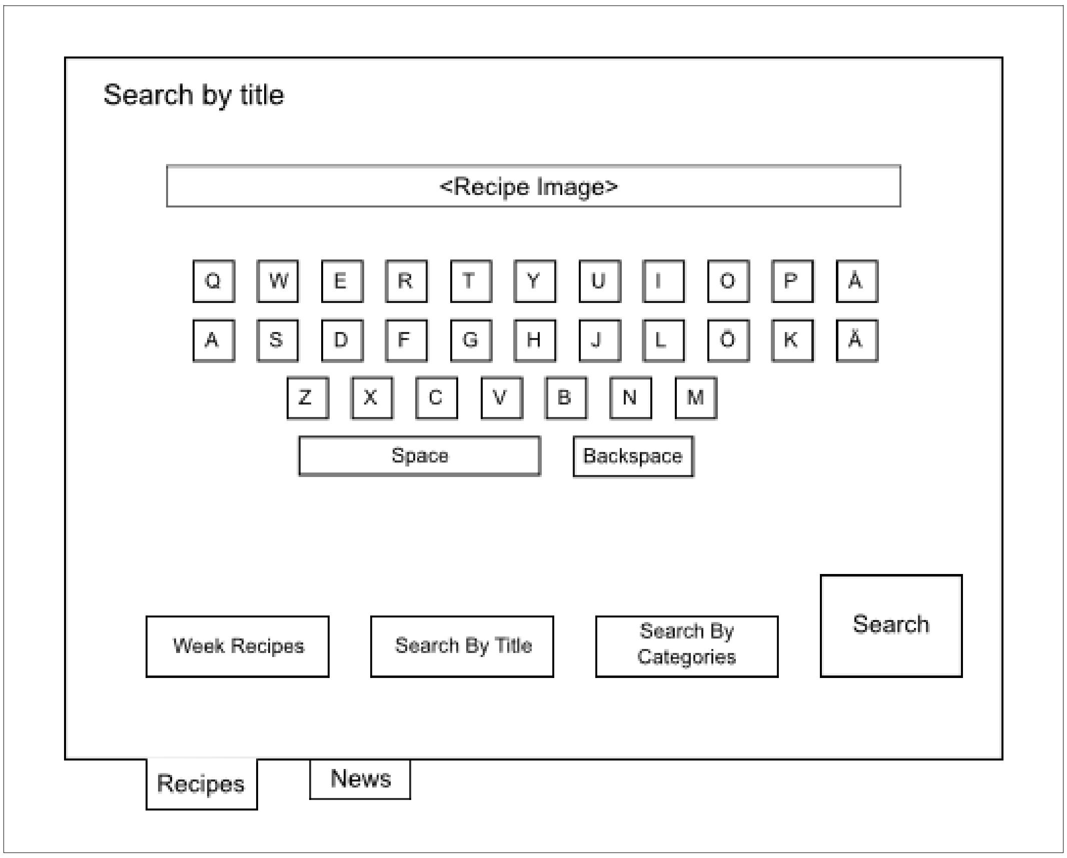


Image 8: Search by title screen

On the top of the screen placed text box control for entering words supposed to be used as search criteria. Because terminal does not have physical keyboard attached virtual analog is displayed under text box. Virtual keyboard allows to enter letters of Swedish alphabet and spaces and also to delete last entered symbol. When user finishes entering search words he has to press 'Search' button for launching search process. After search is done user automatically will be shown search results screen described below or he has to be informed that no recipes were found.

Similarly to week recipes screen on search interface user sees and can use three buttons for switching between search alternatives and week recipes.

Next screen shown on Image 9 is displayed, when user press 'Search By Categories' button. This screen allows to chooses one or several category values as search criteria.

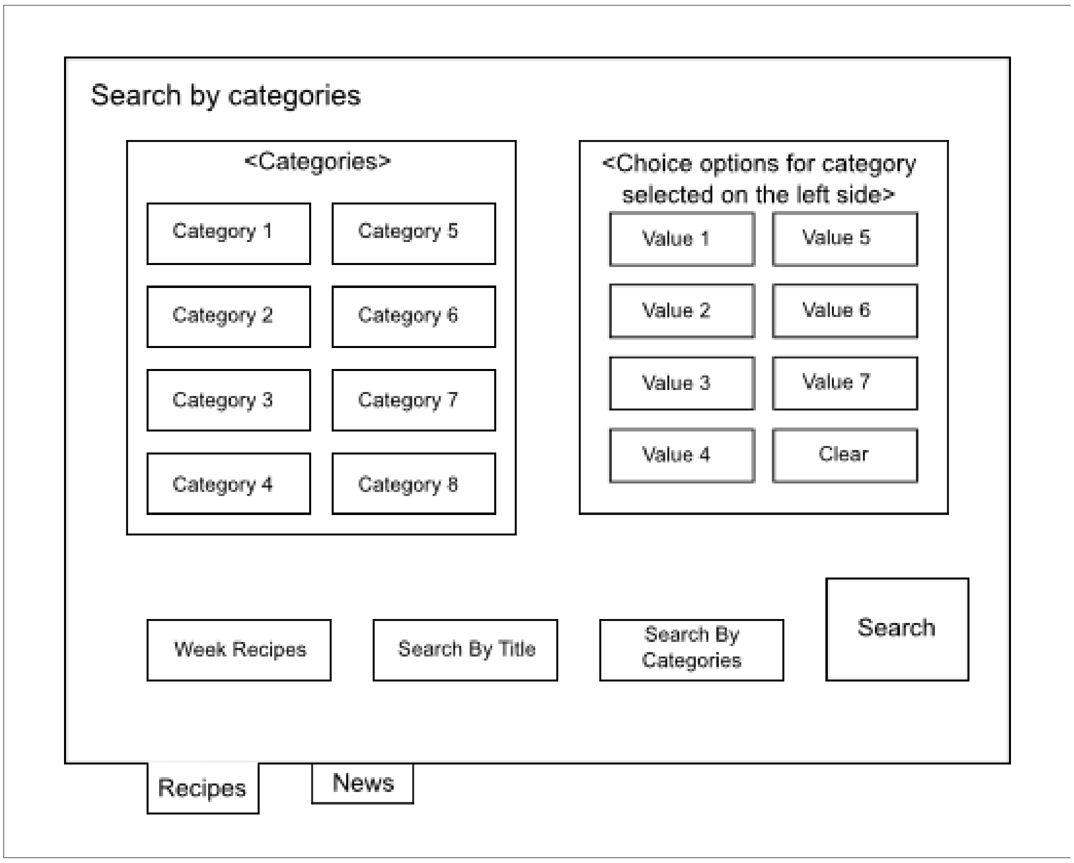


Image 9: Search by categories screen

On the left of the screen user sees a panel with buttons corresponding to different recipe categories that can be used for search. There can be two types of buttons depending on how many values category has. When category has only one value, for example, 'vegetarian' category, corresponding button behave as check-box control. Pressing such button switches choice of corresponding category value in search criteria and appearance of the button changes between two states.

Second type of buttons on left panel are for categories with several values. Pressing on such buttons will lead to second panel with buttons will appear on the right side of the interface. Buttons on this right panel corresponds to values of category chosen on the left panel. When user press one of the buttons related value will be added to search criteria and panel will be automatically hidden. There is also auxiliary button 'Clear' on the values panel, pressing on which removes chosen value from search criteria. One important thing to note is that although recipe can be assigned multiple values of same category, during search user can use only one value of given category in search criteria, choosing another value will automatically replace previous.

It is not shown the screen above, but when panel on the right side is hidden on its place shown list of values currently chosen for search. Also category buttons on the left panel will change their appearance depending on whether value of related category is currently among search criteria or not.

When choice of category values is finished user has to use search button for starting search. Analogously to searching by title, when search is finished user interface automatically shows result or information that search was unsuccessful.

Three buttons on the bottom of the screen have same meaning and behavior as on week recipe or search by title screens.

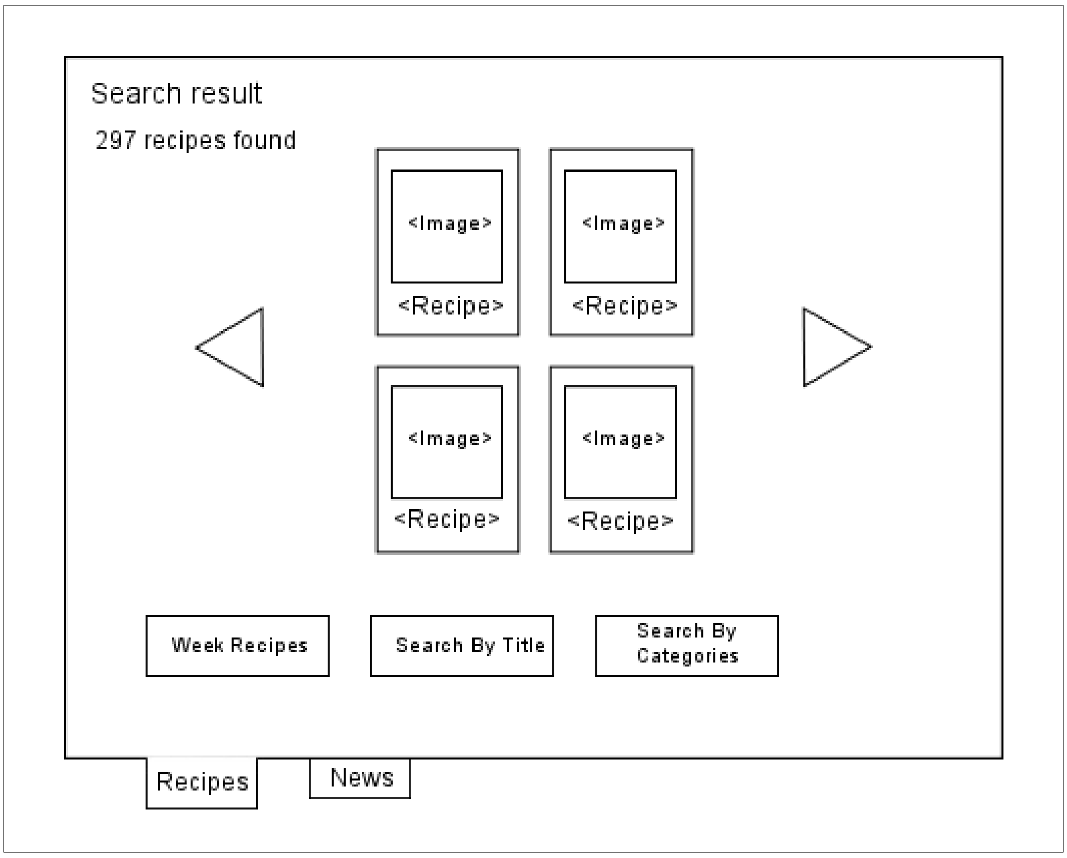


Image 10: Search result screen

As it is mentioned previously, after any of search alternatives is chosen and executed and result is not empty system automatically displays it to the user and blueprint on Image 10 depicts corresponding interface.

In the left-top corner under label with description of current screen user sees number fetched.

In the middle of the screen four recipe buttons are shown. This buttons are made similar way to week recipe buttons. They contain recipe's title and thumbnail of photo, but in this case it is bigger than one on week recipe button.

Number of recipes in result in many cases will exceed four and user can use two buttons on left and right side of screen for moving between recipes in result set.

When user press one recipes button application interface will look as on the Image 11.

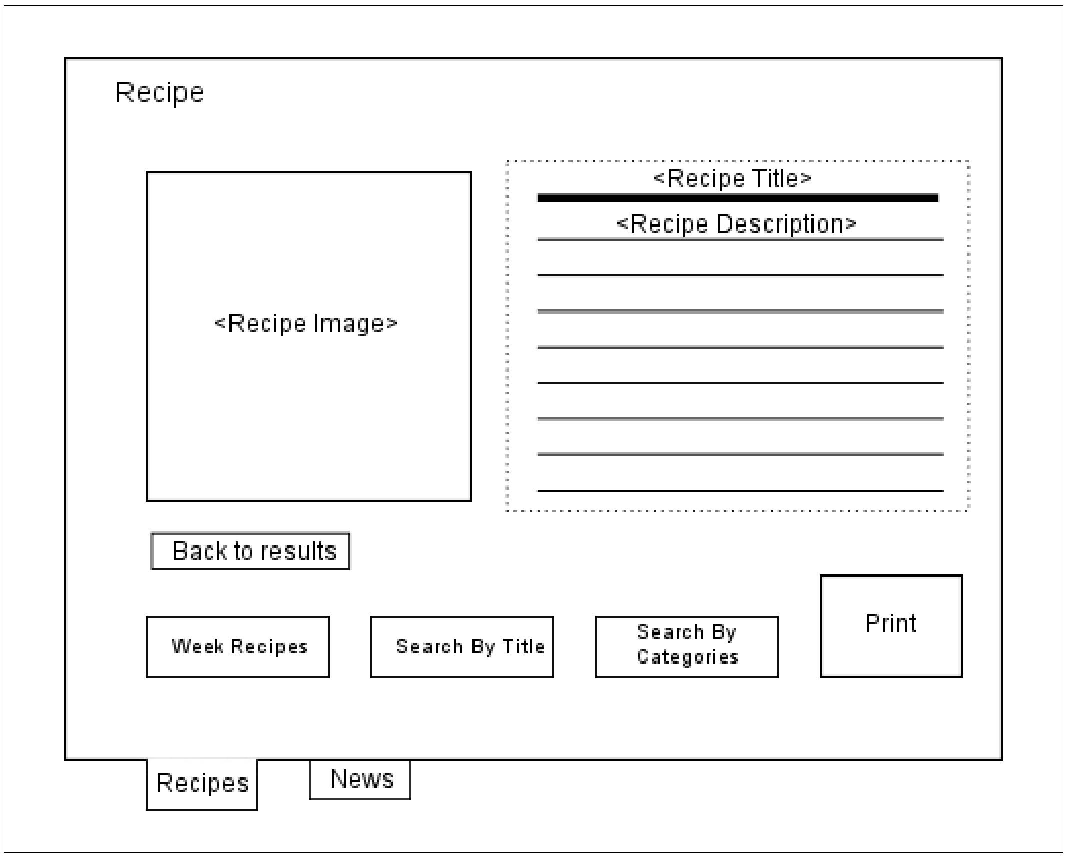


Image 11: Recipe view screen

This screen very similar to week recipes screen. It displays full description of recipe with large photo. One difference of this screen from week recipes screen is button below recipe photo. Pressing this button returns back screen with results of last search made by user – screen from which user came to this recipe view screen. Second obvious difference is absent of week recipe buttons on the right side.

'Print' button have same behavior as analogous button on week recipes screen.

All above described screens are displayed when 'Recipes' tab is chosen. Screen on Image 12 appears when user chooses second tab – 'News'.

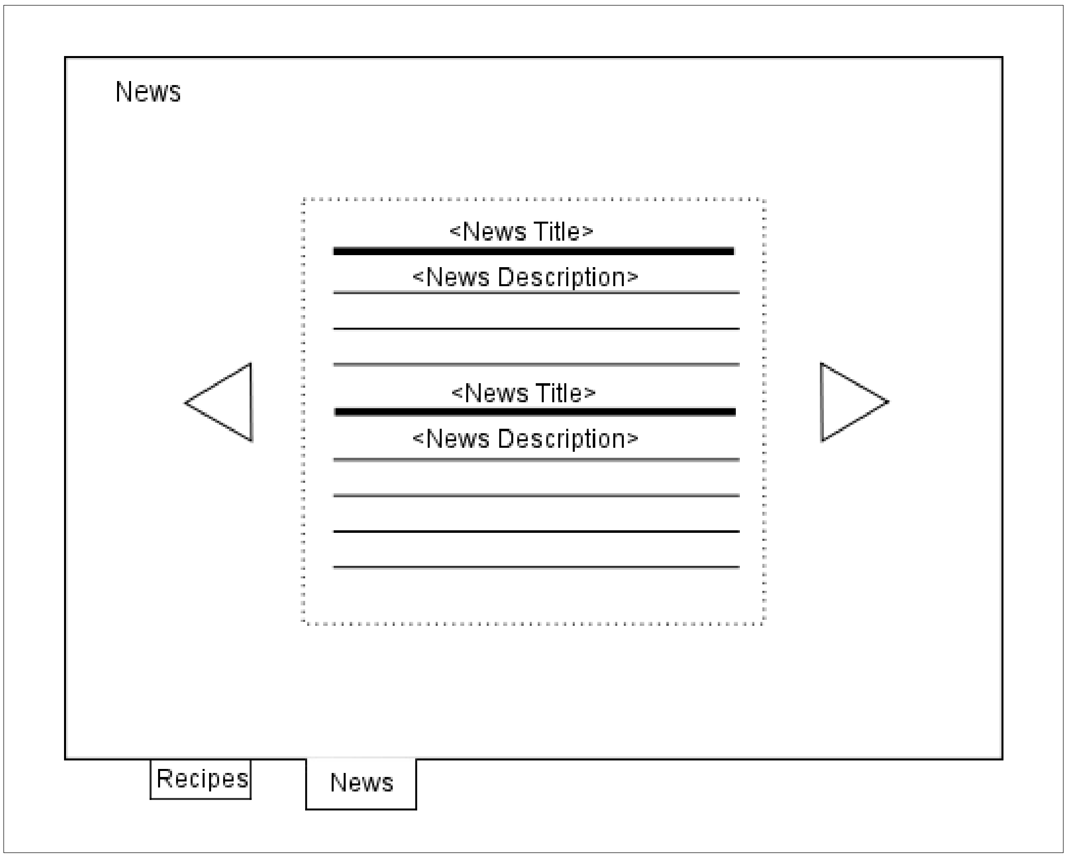


Image 12: News view screen

In the center of screen user sees panel with news titles and descriptions. If concatenated news titles and descriptions cannot fit screen then this text is split to pages and user can use two buttons on the left and right side to move between available pages.

Software components and tools

Modern development of any kind of software systems usually involves using of third-party components, instead of development of the whole system from scratch. Using components allows to decrease amount of time and efforts put in the development process. At the same time it helps to lessen number of errors and bugs as long as stable and proven components are chosen.

The kiosk system is not exception from this general rule. The system does not require any compatibility with obsolete software and choice of components and technologies is free. Besides the obvious criteria of components' choice, such as stability and functionality, developed system requires a components to be free or open-source and have an appropriate license for use in commercial applications. For example, components with only GPL license can not be used in the application.

Second important criterion is the time required to become familiar with new components or frameworks. Besides general programming skills project requires advanced knowledge of web design and development of web applications.

An additional task is to build run-time environment for the application, which includes server- and client-side operating systems, web-browser and other auxiliary software.

Programming language and frameworks

Nowadays developer or development team can create web applications using wide range of languages, frameworks and tools. Some technologies provide solutions for both client- and server-side parts of application, other only for one of the parts. Appropriate choice depends on application's purpose and requirements, skills and knowledges possessed by development team.

One of the essential requirements of kiosk system is that the user interface has to be hosted entirely on the client side and should not employ a full page reload during interaction with user. This requirement narrows number of technologies applicable for the application, but still many options exist. As the user interface is most important part of the system at first client-side technologies will be discussed.

Google Web Toolkit - client side framework

Considered kiosk system can be classified as rich Internet application (RIA) – a web application with many desktop application features. For a development of such applications one can use many approaches and frameworks. In general, they can be divided in two categories: first requires external plug-in for web browser to be able to run RIA; second category don't need external plug-ins and a web application can run directly in almost any web browser.

Plugin based technologies

The first category includes several widely used technologies: Adobe Flash, Microsoft Silverlight, Curl.

The most popular technology from is Adobe Flash platform. It allows to create animated interactive applications with the possibility to play video and audio content. It uses own programming language ActionScript for controlling animations, handling user input etc. Adobe Flash is a proprietary technology and while flash player plug-in is available for free, the original authoring tools are quite expensive. Flash fits very good to requirements of developed application regarding features of user interface. It is a cross-platform solution with plug-ins for the most of web browsers and operating systems. But it has several disadvantages that makes it a poor candidate for use. The first disadvantage is, as it was said previously, is a high cost of the original authoring tool, though there are free alternatives available. The second reason for making Flash unattractive is zero-level knowledge and experience of development team in this technology.

Silverlight is another web framework. It is realised by Microsoft and its purpose is similar to Adobe Flash. Silverlight provides solid foundation for the development of user interfaces with rich multimedia features. Silverlight applications can be written on any of .NET languages (languages supported by .NET: C#, Visual Basic, C++, etc.) It is available for the most of web browsers, but has the original support only for different versions of Microsoft Windows and Apple MacOS, however a third-party runtime environment for Linux available. This approach has same advantages and disadvantages for the system as Adobe Flash technology and lack of experience and knowledge of it make Silverlight less attractive choice.

Curl is the least well-known from above frameworks. Curl is a programming language designed for interactive web applications and targeted mainly for business-to-business applications. It includes both features of the text markup and scripting within one technology. For executing Curl applications on the client side web browser plug-in is required. Curl is a proprietary framework freely available for non-commercial use, but its commercial use for intranet applications should be paid. Due to an absent of Curl knowledge and its commercial nature this framework cannot be considered as an appropriate choice for the developed system.

The all above frameworks share one important advantage and one disadvantage for the developement of the kiosk system. The advantage is the existence of development tools and the possibility to apply software engineering methodologies, such as debugging, code reuse, design patterns, etc.. This is very important for productive and controllable development process. Also these technologies allow to avoid differences between browsers and operating systems as plug-ins behave similar way across different platforms. However having a plug-in turns out to be also a significant disadvantage for the system because it adds complexity during deployment process and plug-ins can be a source of additional errors and problems during development process.

Script based technologies.

The second category of frameworks based mainly on different sets of Java Script libraries. There are many such frameworks and the most well-known are Dojo, jQuery, Prototype & script.aculo.us, ExtJS. All of these libraries allows to use Ajax – a set of web development technologies for building rich client-side front-ends. Ajax usually includes an asynchronous data transmission between client-side front-end and server back-end, complex visual elements and effects for user interface, based on JavaScript and DOM, and an employment of XML for data exchange. Ajax frameworks are independent of operating system used on client-side and usually hide differences between web browsers. Some of these libraries are free for usage in commercial applications, others require fee for it, while third can be used only in open-source projects.

General problem with JavaScript libraries is a lack of good development tools comparable with development environments for other popular languages, such Java, C++ and other. Another problem is that JavaScript is additional language to learn, because usually server back-end is written using other languages, though server-side JavaScript solutions exist.

Google Web Toolkit

About three years ago Google released their own framework for web application development – Google Web Toolkit (GWT). This framework combines best features of both web application frameworks categories.

Web applications developed with GWT do not require any plug-ins for web browser, because usually GWT applications consist of only JavaScript code, HTML and CSS documents and other standard media files (however, any other files can be included and used by application and then additional plug-ins might be required). This feature make GWT more flexible and cross-platform similar to the JavaScript frameworks. The big difference between GWT and many other JavaScript frameworks is that the development process does not involve JavaScript programming, though custom JavaScript code can be added to application. Web applications are created using Java programming language. Then Java source code is compiled to JavaScript by GWT compiler. Resulted code can be optimized during compilation for size and speed and automatically versions for different web browser are created.

Using Java as programming languages allows to use the most of the Java development tools and to apply different software engineering techniques to make development process more controllable and effective.

Additional plus of GWT is that it is licensed under Apache License 2.0. This permissive license allows free usage of GWT for any kind of applications.

After studying and comparison of different web technologies and approaches Google Web Toolkit was chosen for development of client-side of kiosk system. The choice is based on several reasons:

- GWT uses Java as programming language and development team has solid experience with this language in contrast to JavaScript;
- GWT can be freely used for development commercial application;
- GWT includes many customizable widgets for ease design of user interface. It is also easy to create own user interface elements based on standard GWT widgets thorough inheritance or composition;
- GWT allows data transfer between client and server in form of Java objects;
- despite of fact that GWT uses Java for development it easily allows to use HTML and CSS for user interface design;
- web applications developed using GWT can be executed on most of popular browser without additional plug-ins and on all operating systems, where compatible browsers can run.

On the moment of development start Google Web Toolkit 1.5 was latest release, but during development it was upgraded to version 1.6.

Programming language

At the moment when GWT was picked as the client-side framework, Java was chosen as one of the languages. For server-side part one or several other languages still can be used. But having one language during development process makes phases of implementation and testing much easier. That is why employment of any other programming languages should be avoided, except possibly small code fragments on JavaScript.

Due to fact that the kiosk system is web application and client side will run in web browser also HTML and CSS will be used during interface design.

Java Enterprise Edition 5 – server-side framework

The number of technologies for server-side programming existing nowadays is comparable or even exceeds the number of client-sides frameworks. They can be categorized by the language or technology lying on the bottom - PHP, ASP.NET, Java, Python, Perl and many others. Additionally for every category might exist many frameworks or approaches how basic technology can be applied.

The choice of server-side technology in the case of developed system is narrowed by having Java programming language picked for development, but the choice is still wide. The most of Java-based server technologies uses parts of Java Enterprise platform and especially Java Servlet API as common basis. Among these technologies the most popular are JavaServer Faces, Spring, Apache Struts, JBoss Seam. All these technologies includes solutions for both the server logic programming and the user interface producing.

After Google Web Toolkit was chosen as the framework for user interface the kiosk system requires only instruments for server side part. This part does not contain any complex logic and does not require a communication with external services, except database connection and client request handling, that is why Java Enterprise by itself will be enough for the implementation of server-side functionality. Also Java Enterprise is a standard that is why there are many implementations of its specifications and many application servers support it, which provides flexibility and possibility to change some components without rewriting code.

Java Enterprise Edition 5 contains many components useful for wide range of Internet applications: Enterprise JavaBeans 3.0, Java Servlets 2.5, JavaServer Faces 1.2, JavaServer Pages 2.1, Java Persistence API, Web Services APIs and other.

The most interesting for developed system are:

- Java Servlet and Filter API, for handling client requests;
- Java Persistence API, for database information retrieval and management;
- Enterprise Java Beans for separating business logic.

Java Persistence API (JPA) specifies object-oriented way for database access and requires one of object-relational mapping (ORM) frameworks to be used. ORM frameworks hides inconsistent between data representation in relational form in databases and object-orient form in programming language. There are several ORM frameworks support implementation of JPA specifications. Among them are TopLink, EclipseLink and Hibernate. Hibernate was chosen during implementation as ORM framework due large community and ability to handle large objects from PostgreSQL database system.

Application server - GlassFish

Application server is software product required to for execution of business logic components of application. In multi-tier architecture it is usually placed between web server, which receives HTTP requests from client and sends responses back, and database system, which stores application data. Most application servers are part of software products that includes also a web server functionality.

Java Enterprise platform chosen for server-side implementation is standard and that is why there are many application servers, which supports it. Most well-known and widely used Java EE servers are Sun Java System Application Server (Sun JSAS), JBoss, WebLogic, WebSphere and GlassFish. All of them, except Oracle's WebLogic server, allows free of charge use and provide similar functionality.

Development team had previous small experience only with GlassFish server, which forced to choose it for development and production of developed system.

Besides some experience of GlassFish following features make this application server attractive for developed system:

- GlassFish is open-source server ready for production use;
- It is reference implementation of Sun's Java Enterprise platform and much source code is contributed by Sun.
- It has good integration and support of most popular Java development tools: Eclipse and NetBeans, which makes development and testing process easier;
- GlassFish supports clustering and can be scaled-up when load for server grows.

Last stable release of GlassFish is 2.1 and it was chosen for development and deployment of system.

Database system - PostgreSQL

Next important for developed system software product is database management system. Development team had good knowledge and previous experience with different kind of relational database management systems (RDBMS).

For project's purposes standard relational database management system with possibility to store recipe images inside database is needed and there are many database systems that can satisfy this. Most well-known and widely used are Oracle, DB2 from IBM, Microsoft SQL Server, MySQL.

The main disadvantage of above mentioned products as full-featured versions of them are quite expensive and cannot be freely used for commercial applications, but all of them have corresponding free editions: Oracle Express Edition, DB2 Express-C, SQL Server Express Edition and MySQL Community Server. These free versions have following limitations:

- Oracle Express Edition (Oracle XE) is limited to 4 GB of user data and to 1 GB of RAM (SGA+PGA). XE will use no more than one CPU.
- DB2 Express-C has no limit on number of users or on database size and can run on Windows and Linux machines of any size, but the database engine will use only two CPU cores and 2GB of RAM.
- SQL Server Express Edition has no limitations on the number of databases or users supported, but it is limited to using one processor, 1 GB memory and 4 GB database files.
- MySQL Community Server has no limitations, but MySQL in general has some criticism about its performance, stability and support for standard SQL features.

Having above limitations alternative database system was being looked for and choice was made towards PostgreSQL database system.

PostgreSQL is a powerful, open source object-relational database system with full support of ANSI 92/99 SQL standards. It runs on all major operating systems, including Linux, UNIX and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures. It has no limits on hardware components usage and database and table limits. It is also known to have both high performance and high availability and according to benchmarks its performance is comparable commercial enterprise level database systems.

At the moment of choice made PostgreSQL 8.3 was latest available version.

Operating system

All software components chosen for developed system can run several operating systems, such as different versions of Microsoft Windows, Linux, Unix and MacOS.

Microsoft Windows and MacOS do not have free versions or editions, that is why they are least attractive choice for developed system for both server- and client-side parts.

UNIX and, especially, Linux have many free distributives with different set of features and hardware requirements, which allows flexible choice operating system separately for client and server.

UNIX family containing several free distributives, such as FreeBSD, OpenBSD, NetBSD and other, yields to Linux in flexibility of choice especially regarding light-weight version for desktop computers. Because of this choice made to use one of Linux distributives for client part of system and for decreasing cost of maintenance one of Linux versions also has to be used on server.

Client-side operating system - Puppy Linux

Client terminal requires minimal set of software to be run on it. It needs only web browser with graphical interface, ability to print recipes and process touchscreen input and network requests through USB modem. The all software installed on client should fit 1 GB disk space and and be able to run on computer with around 1 GHz processor and 512 MB of operative memory . These requirements shapes choice of appropriate Linux distributive.

Most of desktop versions of Linux has quite heavy graphical interface and large set of pre-installed software packages, because they are intended to be used in normal desktop environments. But there are several distributives with graphical interfaces and low hard disk and memory requirements. Two of them being known most stable were considered as possible candidates. These are Damn Small Linux (DSL) and Puppy Linux.

Damn Small Linux requires only 50MB of free space on hard disk, 16 MB of memory and 486 processor to run. It contains graphical desktop and simple and light-weight window manager and Firefox 2 web browser for running web surfing and running web applications. Disadvantages of DSL are non-standard printer support, older versions of software packages and not supported touchscreen functionality.

Second candidate, Puppy Linux, has higher system requirements. It needs around 128 MB of memory, around 100 MB of free space on hard disk and at least 166 MHz Pentium processor. By default it contains many additional software packages installed, but there are versions of Puppy Linux with only basic software included with possibility add required software later. One of pluses of Puppy Linux comparing to DSL is that it has full X.Org server required for touch screen driver to be able to work. Also Puppy Linux is closer to the normal desktop versions of Linux and offers more flexibility for adding new software than DSL.

Additional advantage of Puppy Linux is that it can be installed in, so called, frugal mode. In this mode a copy of operating system image from installation media is put on hard disk and then during system boot it is mounted in read-only mode and information from it is loaded in operative memory. All configuration updates and additionally installed software are stored in separate file, which is loaded together with Puppy Linux image during startup. Frugal installation allows to keep system files consistent and to easily restore the system in case of failure. Also it can help to decrease number of writes to hard disk and therefore increase life of compact flash storage media.

In the comparison with DSL Puppy Linux looks more suitable for the developed system purposes and Puppy Linux 4.1 was chosen as the client-side operating system.

Server-side operating system - Ubuntu Server Edition

Server part of the kiosk system contains application server and database system running on it. Chosen GlassFish and PostgreSQL can run on almost any Linux distributive. Main requirements for the server-side operating system is availability of command-line interface, remote access for its configuration and preferably absence of graphical user interface for saving processor time, operative memory and hard disk space.

There are several distributives generally used as a server operating systems. They all provide similar set of features and characteristics and with given requirements all are suitable for the developed system. The key factor for appropriate server distributive is experience with different Linux versions. The most extensive knowledge and experience developers have with Ubuntu-based versions of Linux and therefore Ubuntu Server Edition 8.10 was chosen.

This distributive was installed on virtual server on external site and all its management is done remotely using Secure Shell (SSH) network protocol and utilities

Web browser - Firefox

Last component to choose is the web browser for client side. Its choice is determined by the operating system used on client and the number of features supported.

The kiosk system requires a web browser with good support of standard web technologies – HTML 4, CSS 2, JavaScript, DOM, XML and XSLT. Also browser should have possibility to play back audio files either built in or via external plug-in and run Java applets, because printing will be implemented using this technology.

Having Linux installed on the client side the most obvious choice for web browser is Opera or Firefox. Later was chosen as more wide-used and feature-rich product.

Firefox 3.0 was the latest stable version at the moment of decision and in addition to it two components were added: Java Runtime Environment and corresponding plug-in and MPlayer plug-in for ability to play OGG audio files.

Conclusion

As the result of thesis project functional RecipeKiosk system was developed with one kiosk terminal installed as pilot version in one of food stores. All Juzy Systems AB requirements were satisfied and the desired software product was built.

According to the customer wish to decrease the cost of the system as much as possible during development process were found software components, which are free or open-source and allow their commercial application. Found components were successfully applied, used or integrated in the developed system.

References

[1] http://code.google.com/webtoolkit/ - home page for Google Web Toolkit framework.

[2] http://java.sun.com/javaee/ - description and documentation for Java Enterprise platform.

[3] http://glassfish.dev.java.net/ - GlassFish application server's home page.

[4] http://www.postgresql.org/ - PostgreSQL object-relational database management system's home page.

[5] http://www.puppylinux.com/ - community web site of Puppy Linux.

[6] http://www.puppylinux.org/ - official web site of Puppy Linux.

[7] http://www.ubuntu.com/ - official web site of Ubuntu Linux.