SØNDERBORG KØKKEN SYSTEM

Title:
Sønderborg Køkken System

Author:
Kristian Hansen Bock,
Asger Jessen and
Kevin Petersen

Institution:
EASV Sønderborg

Class:
DMU 14

Date:
10/06 2016

Teachers:
KSK, THA, CJO, SBA

Resume

This report details the development and production of a system for Sønderborg Køkken, with the function of creating and handling Orders. The content will detail the 4 major design phases: Inception, Elaboration, Construction and Transition. In the Inception phase the design, planning and requirements are detailed, while in the elaboration phase designs were finished and the work on the product was begun, finally because the project used the agile development method “Scrum”, the final 2 phases construction and transition were repeated one after another as tasks from the backlog went through it. Finally, after detailing the work in the 4 phases, the Design choices are described as well as coding choices made in the product, followed by the future improvements that can be made and the conclusion of the report.

Indholdsfortegnelse

Introduction

The purpose of this report is the document the development of the Sønderborg Køkken system, including choices made during the planning and production phases, as well as explaining how the product completes different actions and if special code choices have been made and why. The reason this project was started was to develop a system to handle orders for Sønderborg Køkken digitally in order to replace their use of paper documents that were being passed around as stations completed their tasks.

Glossary

OC - Order Confirmation.
OO - Order Overview.
DB - Database.
MVC - Model, View, Control.
UI - User Interface

Project Description

The purpose of the project is to create a system for handling and displaying order information and related files to the production team at sønderborg køkken. The displayed information should be shown on tablet devices connected to the office through some means in order to receive information and share it with the other members and users.

Project Requirements

- Back-end must be based on VisualBasic.Net and or C#
- Front-end must be either Android or a web application with the angularJS framework.
    - Communication between back-end and front-end highly recommended to be a restful web service.
- Minimum 80% of the technologies must be those that have been taught in lessons.
- System development choice is free.

Development methods

Scrum

The main development method used for the project is Scrum, which is an agile system development method. While not designating a scrum master, artefacts and methods from Scrum were used. Some used artefacts were a product backlog, kanban board and burndown chart, the last of which was in the end scrapped.

XP

Extreme programming was used in the project. Tasks were kept small and compared to the value as described by the customer and designs were kept simple in order to quickly complete as much as possible. The pair programming was also used on some tasks to improve the quality of the code work.

Inception phase

In the Inception phase the project idea is generated, minimum requirements for the product in the form of a product-backlog and the technologies to be used are formulated and the feasibility of the project is researched.

Time estimation

The time to make the system is 4 work weeks and 1 work day. This is 21 days spending 6 hours a day, which is 126 hours. With weekends which the group will also use it is 29 days or 174 hours. Since there are 3 people in the group working on the project it is 522 man hours, which should be more than enough to produce the system and accompanying report.

Feasibility study

The customer is a company that has a design department who through a middle link want to electronically send information to the production department. The wishes are to create a system that holds order information made in the design department, and allows the production department to view the information and accompanying files.

This feasibility study focuses on the creation of a system that achieves the goals of the customer.

The time is estimated to be more than enough to finish a working program. The resources needed are access to coding languages and database services in order to release the system. As students the team has access to coding tools that usually require a paid professional license, and the rest of the services have free to distribute licences, for that reason tools are not a problem.

There are a few architectural solutions depending how the options are mixed, however the biggest differences are on the production side.
Given the requirements, there were 2 project solutions:

- The first one is to have the mobile devices connected to the website through an angular web service held on an intranet connection
    - Pros:
        - User Interface (UI) can be created to be as pretty and usable as possible with almost no effort.
        - Web services are easy to upgrade and work across multiple devices as they all read it through browsers.
    - Middle ground:
        - Solution is secured by the company's local wifi connection
    - Cons:
        - The team didn't see a good solution done with javascript.
        - The service is not usable if the connection to the web service is unavailable.
- The second one is to have the mobile devices connected to the web service through an Android app installed locally on the device
    - Pros:
        - Solution is secured by company distribution of the android app.
        - App can hold local backups if the connection to the web service is unavailable.
    - Cons:
        - The Android app is locked to the devices who run on the Android operating system, this means no Apple products unless specifically setup to run Android virtual machines.
        - App takes memory on the device, so it requires expendable space.

Feasibility conclusion

The project is feasible, the team has enough man hours, resources, and knowledge to deliver a working implementable prototype.

Vision

Our vision for Sønderborg Køkken.
The group's goal is for the office employee to upload the e02 file together with the drawings and requisition list. When the files are uploaded, the system will read the e02 and create an OC (Order Confirmation). When the OC is created, it will be stored in the database, it will show up on the production tablets. The production employee can then open the app on then press on an order in the OO (Order Overview). The stored OC will be loaded from the Database and the employee will be able to see the elements and then depending on the station the employee is able to mark it as he begins working on the element, as the employee places the first mark it will show on the OO that the station has begun work on the order. When the employee has filled out his column with mark's it will mark the OO as done. When all the stations have completed an order and all the marks are checked, the order will notify the office that the order is ready for shipment.

Usecases

1. Office
    a. User
        i. Upload e02.
            1. User Clicks the browse button for the e02 file.
            2. A window opens up and User searches for the e02 file.
            3. Double clicks on the e02 file.
            4. System return to the upload tab.
            5. User clicks upload button.
            6. Systems opens a window telling the file is uploaded(or already that the file already exists).
        ii. See OO.
            1. User clicks the OO Tab.
            2. User can see the progress status on the Order on the various stations, and see how many notes there are.
            3. User double clicks on the order number.
            4. a windows pops up.
            5. user can read the note(s).
2. Production

. User

. Get Orders
    1. User enters either of the 3 activities.
    2. Activities create a connection to the service.
    3. Service sends list of orders.
    4. Activities receive orders and pick out information as needed.

i. Get Update point
    1. handler establishes a connection to service and requests getUpdate.
    2. Service sends its own number of updates.
    3. The number of updates from the service is compared to its own number of updates to determine if there are new updates.

ii. View order confirmation
    1. Order is clicked on the OO activity.
    2. OC activity is started.
    3. OrderResult is retrieved and desired order is picked out.

iii. Update element status by station
    1. User opens OC activity for chosen order.
    2. User clicks button to update desired station.
    3. Station status is inverted of current status.
    4. If all stations are “Done” order status is set to done.

iv. View Notes
    1. User clicks notes button on a desired order in either OC or OO activity.
    2. The NotesView activity is opened.
    3. The notes are picked from OrderResult and added to a list.
    4. The notes are displayed.

v. Add Note
    1. In NotesView user clicks on the text field.
    2. The keyboard is shown.
    3. User inputs new note.
    4. User Clicks Post note.

5. The note is added to display list and update list.
6. The update list is sent to the service through the handler.
7. The display is updated from the display list.

Architecture candidates

Solutions are separated by the implementations of the production team's parts

Solution 1: Angular web implementation

- Office client: either Visual basic or C# based
    - Either language only has minor differences in coding, such as C# is easier to build upon, but is still getting new releases, while Visual basic is a retired programming language, which means that while it is still used by many people, it is no longer supported by the developers.
- Website: Javascript and HTML
    - Harder to check for errors, but easier to keep online.
- Production client: Angular web implementation
    - Web accessible

This solution uses a web solution, which means it's an always-on webpage implemented either on the web or on the company intranet. This solution requires a constant connection when issuing updates, since data might otherwise be lost. However this solution does support more devices, including Apple products and other products.

Solution 2: Android app implementation

- Office client: either Visual basic or C# based
    - Either language only has minor differences in coding, such as C# is easier to build upon, but is still getting new releases, while Visual basic is a retired programming language, which means that while it is still used by many people, it is no longer supported by the developers.
- Web service: either Visual basic or C# based
    - Either language only has minor differences in coding
- Production client: Android application
    - Locally installed

This solution uses an Android based solution which is locally installed on a device and can support local storage for added redundancy in case of web service connection loss. The solution isn't implementable on all devices, since Android doesn't run on Apple products and most computers, without an emulation software.

Candidate choice

The team has chosen solution 2: Android app implementation, with office client and web service both coded in C#. Solution 2 was chosen, because it supports local storage and avoids use of javascript, which is hard to error check compared to Android apps. The choice to use C# is based on the fact that C# resembles java is base code structure, it's a modern programming language and is easier to expand upon, while the simplicity of visual basic is a coding language the team doesn't have as much experience with and it is a retired programming language.

Elaboration phase

During the elaboration phase the designs are finished and work on the product is started, while the vision and time estimation is refined.

Iteration

The project had a single Sprint containing three iterations.

The first iteration: During the first iteration, the group started on the parser for the e02 file and the design. The product backlog for the project was also created during the first iteration, in order to create an overview. The product backlog shows the important tasks for the programming aspects, and the business aspects.

The Second iteration: The group worked on the SQL for the database. Created an UI for the office so that it is possible to send the e02 file to the server and then parse it to be stored in the database. The group tried to get the connection with a different connection methods, some being lower leveled than others. The group had problems sending files, for instance they didn't get sent separately but all in one file instead, though this could have been fixed with a lot of work and time. As the group was limited on time, they decided to create a web service instead, but that included dropping the idea of sending the drawing and the requisition over.
The Android UI took shape and got the app to look like the drawing that was received from the customer. The group had various problems with the sending of files and information to and from the Android app.

The Third iteration: The upload UI for the office is completed and it's possible to upload the e02 file and show the status on the upload, the OO is also done for the office site, the group can now check how far an order is done, the group can see which station has started on which order and see if they are done. The web service part was completed.

Refined vision

The program can upload the e02 file onto the REST WebService, and convert it so that it is stored in the DB and creates an OC.
When the order confirmation is created, it will show up on the office Overview as well as on the OO on the Tablet.
Then able to see the order in the OO and on the table, the customer is able to press on the order and it will open the OC that has been created on the server, on the tablet it is possible to place a mark on the various station and as soon as one mark is marked, the OO will show it as the station has begun on the order and when all marks are marked it will show begun and finished.

Refined time estimation

In the original time estimation, it was estimated that all goals set would be reached within the allocated time, however with changes to design choices it is estimated that a few more days are needed to reach the same goals. 35 days with weekends resulting in 210 hours per man or 630 manhours in order to finish the work that was planned. However in order to finish the product to a commercial quality and set it up, the group would likely need another 2 months or more, which would mostly be spent on improving quality, security and design of existing parts, as well as adding extra choice capabilities and capabilities which were cut as their value was deemed too low to bother with during this project.

Error list

Run Time Errors

Office

If the user presses upload before browsing for a file it will give a path empty error.
If the Office user types the path in the text box next to the browse button or after the browse corrects in the text it will give a path error.
if there is no connection and you try to upload a file it will show a message saying it failed to upload.

Android

If the ip address can't be reached, then the tables for the OO and OC will not be created leaving a blank page on either of them. For the NotesView it will not pick out the old notes from the order, but new notes can be written and displayed for the application. If the connection times out, for example when a request is put on hold by the web service, the Android app will stop working and eventually crash unless it is resumed in time.

Construction phase and Transition phase

The Construction phase is the phase in which the majority of the work on the product is made, product parts are developed and once they reach the testing they are sent to the transition phase to be tested. If they have too many errors they are sent back to the construction phase, if they are cleared for release they will be readied for the release they will be part of.

The transition phase focuses on the delivery state of the product. This is done through extensive testing and bug fixing to get rid of defects that are detrimental to the operation of the product. The time spent in the transition phase wasn't long since most of the product was tested throughout the earlier phases.

Design choices

Technologies

C# office

C# is a modern, general-purpose programming language created to use the .NET Framework by Microsoft. The language has a close relation to the other C-languages as well as java, sharing many functions and implementations. C# is used as the main language used when creating business applications, specifically because it excels in working with web services and database stored data.

The team has chosen visual basic, since it resembles java, which is the core of the team's programming experience, and also because it is an industry standard language for the type of system being built. Visual basic is less controlling in the required conventions of methods, but it also makes it too simple and different from the java experience.

The office client will act as a UI for the office to upload orders, blueprints, requisitions and customers to the system, as well as create the order confirmations and get an overview of unfinished orders.

C# Web Service

The Web Service that the group chose is running on C# and which is the same as the office part is using, which allows for easy serializable objects.

The web service the group chose to use was RESTful web service, which use HTTP to send requests and responses using GET, PUT, POST, DELETE. It makes the service a good middleman for the clients on different platforms to get response from the RESTful, getting data from a data source. The response data is either XML or JSON, which is structured in a way that many different programming languages can parse it to objects.

The web service contains the database handler, which is used to store and fetch the order confirmation. The database used is the Microsoft SQL Server, which is linked to the web service, for easy setup in the program. The web service calls the database by stored procedures, which then fetches or stores all the necessary data to create an order confirmation.

Android

Android is an operating system used on the majority of mobile devices, such as tablets and smartphones, it is the best selling OS for these devices and has been since 2013. The system is built on a C core and c++ for added functionality with java used for the UI. The programming language highly resembles a customized java platform. It uses the same syntax and method setup, but has many libraries that are different from the ones used in the java SE, an example would be

the UI, which doesn't use JComponents, but a library similar to swing. Android also has its own virtual machine, which executes the bytecode instead of java code.

The team chose an Android solution over an angular web solution, because it allows for a locally installed software, allowing for easy usage and because it is the go to coding for mobile devices like the tablets and smartphones the customer requested. There is also the possibility for easy backups in the case of lost connection the the web service.

The Android app will be used to retrieve information on active orders, including their blueprints and requisitions from the web service, as well as keeping usable backup of retrieved data and updated data for the orders on the devices during web service connection downtime. The app will update status and comments for orders as the production team progresses on them, for that it will send information to the web service.

Code choices

Code Choice - Web Service

Architecture

The web service is the Controller of the MVC (Model View Controller). The database, including the classes that the OC is build of, is the Model. The group chose to use RESTful service as it matched the groups needs for functionality to communicate between a multi platform/programming language system. The RESTful also offers easy method access between the Service and the Office client.

OrderParser

As one of the main requirement of the product, was to be able to read the Winner Tool file (E02) into the program, which would help the office a lot by having the order confirmation information stored on the service, for easy sharing with the Android and office client using this system. By storing this on the service, makes it possible to add additional data to the order confirmation, such as: Ticking off stations if they've started or finished their part, and adding extra information, like the measurements in square meters of the kitchen they are working on.

The order parser reads the E02 file line by line, identifying what the like contains by the starting number of the read line. The data of the lines are split by the semicolon character, then specific splitted parts are used depending on what the number in the beginning of the line was. Then creates all object parts that defines the order confirmation.

RESTful service

The RESTful service is how the communication point of the program, where all the client can POST or GET from the database using the Database Handler, through Operation Contract

methods. The RESTful service is the method that actively runs in the background listening for call on the methods.

The Android client is calls the method through a URITemplate that is determined in the WebInvoke attribute of the method. The Office client calls the method through a service reference, which works like simple method calls through an instantiated service client. It was impossible to get values through GET methods on the RESTful service that uses URITemplate. To fix this problem, the group decided to create a second RESTful service that was dedicated to service reference GET method calls.

Database Handler

The database handler takes the order confirmation object that was created by the order parser. The database then creates a connection and it constructs the Stored Procedure, that will create the order on the database, and fills the parameters of the stored procedure. The stored procedure will then ensure that the parameters are put in currently to insert the data. The method that creates the order calls the create category method internally, and the category does the same with elements of the order.

Code Choice - Office client

Architecture

The architecture used in the project is a MVC(Model, View, Control) structure, the one used in the office client part is a View, with a tiny bit of model.
the View part of the office is that it presents the order which it gets from the web service
The model part of the office is that it stores the objects of order.

Upload

The main plan for the upload on the Office side was that the group wanted to send the e02, drawings and the requisition files over, to a file server that made, it was supposed to save them so that then links for the drawings and the requisition where from the same place and not from different machines.
The group dropped the Socket connection because when the files were sent over they always arrived as one file instead of separating them into 3 different files, it was decided that it was too low level languages, it was decided to use restful instead and that's what being used now, and the group didn’t want to use more time on the PDF files and now only send the e02 file.
The group chose to have the upload of Blueprints and requisition on the office GUI, so that the clients didn’t know anything and only had to send files to the server, and it would then receive the information that it is send with the status box showing, Green for sent , red for failed and blue for idle, not sent yet. The group are sending the e02 file as a string over to the REST Web Service

Order Overview Office

but the office also has another gui that shows them the OO so they can see the orders and the status on the various station on the orders, if for instance the Station 4 has begun working on a specific order and see if it is done as well as see how many orders station 4 is done with.
The group receive the information about each order as an object and it will update all of the orders as soon as there is an update on the database.
The office receives a JSON from the web service and converts it over to an objects and extracts the needed data from it.

Code Choice - Android

Architecture

The architecture in the Android application is based around the OO. From the OO you can select to view a specific OC or view and make Comments to a specific order. In the background there is a Resthandler which handles connecting the app to the other parts of the system and is used by the update method to keep the displayed views up to date or send data to the system. The Resthandler and the three main activity classes use an Order Result object, which contains any number of orders picked out from the server.

OrderOverview

The main activity of the Android app is the OrderOverview, from there it is possible to select OCs and comments for the different active orders. The orders are sorted into a table of orders which shows the name of the order, which is clickable to get to the OC for it and the progress of the different stations, as well as a button for the comments.

On startup the Activity creates a RestFulHandler object, and executes the update method, which runs in the background and calls itself repeatedly after a short sleep time. The update method tells the handler to connect to the service and check for updates and if there are any it will retrieve an OrderResult object containing a list of orders. then the OrderOverview activity will build a table of orders through the use of a function called build table that uses switch cases to pick out the right info for each column in a row. The Activity uses a tableview inside a scrollview in order to dynamically create unknown numbers of order rows and scroll through them if they take up more space than the size of the screen. Because the update thread runs in the background, there are methods called to update the UI on the main thread, since the background threads cannot update the UI. The Buttons are coded to have their own unique ID so that they don't interfere with each other or end up with the same ClickListeners. The ClickListeners for the 2 types of buttons are coded in their own methods that retrieve them, since Android doesn't allow the code to add the ClickListeners inside the switch cases. When clicking one of the buttons the ClickListeners are coded to start a new intent depending on which of the 2 button types are clicked, which then opens a new activity of the appropriate type, while sending along the order name of the selected order.

OrderConfirmation

The OrderConfirmation activity works the same way as the OrderOverview activity except, when it is created it saves a order name that was received from the OO and when updating is uses the order name to pick out only the order that matches the order name. Also instead of having mostly textfields the buildtable function also creates a category row with the category name for the following elements and a show/hide button which hides the buttons in the table so as to only show information for the order. The Station fields from OO are also replaced with buttons for updating the status, as of the current time the updating of element status to the rest service has not been implemented yet.

The ClickListeners for the show/hide button, works by using a list of all button id's, other than the show/hide buttons, which are created when the buttons are created and added to the list. The method iterates through the list and hides the buttons if they aren't already hidden, in which case it shows them again. The way they are hidden is using the "GONE" method instead of "INVISIBLE" because that way the space is reallocated to the remaining fields when the buttons are hidden. The updatestation function only updates the status on the OC, but it doesn't send the update to the service, it checks the current status and changes it to the opposite, completed or not completed.

NotesView

Unlike the other 2 activities the NotesView doesn't build tables, but picks out the notes from a specific order and sorts them into an arraylist which is then used to create a listview with them on it. When it is created it receives an order name and loads everything from the order. It does not have an update method for continuous updating unlike the other activities, but it does have a method to post new comments to orders on the rest service, which are handled by the resthandler. When the upload to the server is complete, the newcomments list will be emptied.

RestHandler

The resthandler is a standard java class used as an object. When the class is instantiated the base url is set to the get orders address for the hard coded ip address. If the readStream method is called, the connection is setup to GET information from a restservice using a stream reader and inputstreams from the urlConnection class. Using the reader and the expected dataobjects the gson class converts a string read from the server from json into a getUpdate object holding the current update count, if said update count is less that the handlers updatepoint variable, then it will repeat the methodology into an OrderResult object, which is then returned. The updatepoint variable is hardcoded to start at -1 in order to always get an object the first time after the creation of the handler and will be set to the value of the getUpdate object. If the handler is up to date on the updates, it will just return a null object, which will be ignored in the receiving classes. The writeStream method established a POST connection and converts a string into a json object before sending it.

OrderData

The order data is stored in 2 different objects, one of which contains multiple other objects and arrays. The first object is the getUpdate object which only holds an integer variable and is used sync the OrderResult on the application with the service stored OrderResult. The OrderResult object holds an arraylist of Orders, which hold basic information on the order like it's name, number and due date, while also holding objects containing the Status of the order and 2 arraylists containing Notes and Categories, the last of which holds an arraylist of the elements belonging to the categories under the order, with every element having descriptions and their own boolean array for the element progress.

For now most of the objects are used, though certain variables haven't been implemented in the rest of the app so they only take up space until that point.

Code Choice - SQL

The SQL that is made creates tables and stores procedures in order to create the order confirmation from the e02 file.
The SQL creates five tables, the order of the creation of the tables is specific made since the e02 file contains the Company information, which is used before you can create an order, and the order need to be made before you can add Note table, Order Category table, which contains the Order elements table.
With that said, the creation of the stored procedures does not have to be made in any specific order, but it is good manners and order to create it in the same order as the tables.

Future improvements

Implementation of lacking product backlog

Updating the status of orders both on the Android and Web Service side needs to be implemented, specifically the method calls from the webservice to update the DB and the connection and sent data from the Android application. The Android application also still lacks a local storage method, even though it is the clearest advantage it has over an angular Web Service. A way to implement this is to use locally stored JSON files on the application, which are loaded at startup and then compared to the getUpdate from the services if connection can be made, otherwise it is automatically used and saved regularly during the online use. The last major method that hasn't been implemented for Android at all is the update the overall status of the order from active to done, once all stations in all elements have been marked as done automatically and then update it to the server.

The office side also lacks the update status for the various station from the DB, as well as the upload of the Blueprint and Requisition PDF files need to be implemented so that the files are send to the web service, and a link will be a reference to the PDF files so that they can open them on the Android and on the Office client.

Implementation of new methods

The edit of an order delivery time needs to be implemented, so that when a customer needs to delay the order, the Office user can go into an order and edit the delivery date.
the office should also be able to type in the M^3 size of the shipment and the amount of elements.
For Android the new methods that hadn't been planned are methods of updating the NotesView continuously the same way the other activities do in the background, as well as creating a method for keep everything in order when the screen is rotated, as Android has a habit of rebuilding the activities whenever it rotates the screen.

Quality enhancement

Improve code to safely guard against type errors, invalid argument errors, null point exceptions as well as improving efficiency of existing code and UIs.

Security implementation

The data going from and to the service could be encrypted if it really is necessary, but seeing as the data that is being sent forth and back on the web service is not containing personal data about the customers or bank information, so the need for encrypting the data and security is minimal at most. Furthermore a login method could be setup, with login information only having to be entered the first time a user uses the application after the installation in order to minimize the annoyance in remembering

Conclusion

Going through a lengthy design phase, which took into consideration many requirements, technological decision and detailed multiple use cases, production was started following an agile development cycle known as Scrum. The phases the development went through, were the inception, elaboration, construction and transition phases. Many code choices were carefully considered, tested and changed throughout the development as some better alternatives either showed themselves or were needed. However while not all planned tasks were completed, the product is still in a deployment ready condition with multiple planned enhancements and added functionalities.