Western Norway
University of
Applied Sciences

BACHELOR’S THESIS

WA Interview booker

Emilio Samano Hoftaniska,
Jannecke Sundal Nyquist,
Torstein Winjum

Computer engineering/IT
Faculty of Engineering and Science
Supervisor: Richard Kjepso
Submission Date: 03.06.2019

BACHELOR THESIS TITLE PAGE

Report title: WA Intervju-booker WA Interview booker	*Date:* 03.06.19
Authors: Emilio Hoftaniska, Jannecke Sundal Nyquist, Torstein Winjum	*Number of pages:* 37 *Number of appendix pages: 5*
Education program: Computer engineering/Information technology	*Number of CD's/DVD's:* 0
Program contact person: Richard Kjepso	*Classification:* Open
Remarks:	

External company: Wide Assessment	*External company reference:*
External contact person: Andreas Hammerbeck	*Phone:* 97661466

Summary:

This bachelor thesis describes the planning and execution of a development project for Wide Assessment. The goal of the project was to expand the wa.works recruitment platform with an interview booking system.

Wa.works is a web application that serves as a resumé tool for job applicants, and as a screening tool for companies. The platform aims to help companies to find the right candidate, and to help applicants with exposure to companies. The motivation behind the interview booking system was to further facilitate the hiring process by making it easier to invite potential candidates to interviews.

The result of the project was a functional interview booking system, which Wide Assessment plans to integrate into their platform after necessary modifications have been made.

Keywords: ASP.NET Core, React, Typescript, RESTful API

PREFACE

This bachelor thesis describes the planning and execution of a development project for Wide Assessment. The goal of the project was to expand the wa.works recruitment platform with an interview booking system.

The group would like to thank Wide Assessment for the opportunity to work with them on expanding their platform.

Special thanks to Andreas Hammerbeck and Viljar Rolfsen for teaching us the ropes, and for patiently providing technical assistance throughout the project.

Thanks to our internal contact person Richard Kjepso for concise feedback.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Goal, motivation, and context

The hiring process is difficult and time-consuming, both for the applicant and the company looking to fill a position. Wide Assessment facilitates this process, specifically for the IT industry with WA.works. WA.works is a resume and screening platform that lets candidates list their technical skills, and lets companies uncover and compare candidates.

The goal of this project is to develop an interview-booking system and integrate it into the WA.works platform to streamline the hiring process further. The interview booker will let companies invite candidates to interviews directly from the platform and add the interviews as events in a calendar.

1.2 Limitations

In this project, the group is restricted to the programming languages and technologies that the current WA.works platform is built with.

The group must use React for the front end, Redux for state handling, .NET Core for the back end, and open API's from AddEvent for adding events to a calendar.

1.3 Resources

The most important resource for this project is Wide Assessment. They will provide useful information and help with the setup of the work environments.

The group will be using Typescript and React.js for the front end, C# (.NET Core) for the backend, and Docker for deployment. In addition, the use of open API's from AddEvent is required. For managing work, Zenkit for Kanban board and a simplified version of the Scrum model will be used, and Google Drive will be used for sharing and collaborating on documents. A private repository on GitHub will be used for collaborating on the source code.

1.4 Organization of the report

The report consists of the following chapters:

Chapter 1: An introduction to the project.

Chapter 2: A more detailed description of the project and its context.

Chapter 3: The design of the project, discussing different approaches and planning.

Chapter 4: An overview of the planned design of the solution.

Chapter 5: A detailed description of how the solution was implemented.

Chapter 6: The evaluation of the project.

Chapter 7: A discussion of the result.

Chapter 8: Conclusions drawn from the project.

Chapter 9: Literature and references.

Chapter 10: Literature and appendices.

2 PROJECT DESCRIPTION

This chapter is a brief description of the project owner, Wide Assessment, and the project itself. WA's preliminary work on the project will be explained, along with a more detailed description of what is expected of the final product.

2.1 Project owner

The owner of the project is Wide Assessment AS. Wide Assessment started as a project by Gyril Norway AS in 2016 and became its own startup company in 2017. The company uses a recurring SaaS (Software as a Service) revenue model and offers an IT-recruitment and growth platform, WA.works.

Wide Assessment was founded by Stine Andreassen, the CEO of the company, and Arve, the CSO. The company is based in Bergen, Norway, and consists of 10 employees.

IT is a rapidly growing business, with many fields of expertise. Wide Assessment's platform offers a way for companies to find and hire candidates with the specific skills that they are looking for. It is also a platform for candidates to showcase their skills.

The platform allows a candidate user to select their knowledge from an extensive list of technical skills, and rank them by experience. A candidate can also highlight their specific skills, link their skills to projects, state their willingness to move

A company can then post listings for jobs that require a certain set of skills, and get a list of candidates that match, ranked by how well they match.

2.2 Previous work

Wide Assessment has already done preliminary research on which technologies will be used in the project. They have come to the conclusion that making their own calendar and video conference services will be unfeasible. Instead, third-party systems and an in-platform management system with links to the external services are to be used.

2.3 Initial requirements specification

The requirements from Wide Assessment regarding the front end part of the project are that it is to be developed using Typescript with React.js and to use Redux for handling the state. The requirements regarding the back end are that .NET Core framework and PostgreSQL will be used for database queries. Wide Assessment also requires that the group uses open API's from AddEvent, a "Add to Calendar" service, to add the interviews as events in the user's calendar application.

2.4 Initial solution idea

After an initial meeting with Wide Assessment, the following plan… something:

1. **Learn React:** React is the typescript library used in the WA front end. The group had little experience with this, and learning it would be the first order of business.
2. **Become familiar with the front end:** A basic understanding of the structure of the existing front end was necessary to start the project.
3. **Make a modal in the front end:** The first part of the implementation would be to make the visual modal without any data interaction.
4. **Make a controller that can feed dummy data the front end:** Make the modal able to read data from the back end.
5. **Make the controller receive data:** Make the modal able to send data to the front end.
6. **Implement the required database tables:** Additional tables in the database would be needed to accommodate the new interview invitation data.
7. **Make the controller communicate with the database:** Change the controller to actually interact with the database instead of using the dummy data.
8. **Implement a template feature:** Change the modal to be able to save and load interview invitation templates.
9. **Implement datepicker and misc:** Add a better date selection interface, implement other quality-of-life features.

3 PROJECT DESIGN

This chapter describes the planning of the project and discusses different approaches.

3.1 Possible approaches

Wide Assessment specified that they wanted their platform's functionality extended with the possibility for companies to invite candidates to interviews as a result of this project. Wide Assessment also specified that inviting a candidate on the platform must generate and send an email to the candidate. They further specified that the group should use AddEvent's API's to let companies add scheduled interviews to a calendar application.

After becoming familiarised with the existing implementation of the platform, as well as with the React and .NET Core frameworks, the group mapped out two viable approaches to the project.

3.1.1 Dummy create-react-app approach

In this approach, the group would use the create-react-app tool to set up a mock React application where the components needed for the front end part of the interview booker are initially implemented. When the components were complete, the code from the mock front end application would be copied into the actual front end application of the WA.works platform.

After integrating the components in the platform, the group would make its part of the WA.works UI able to receive data from a controller, which is implemented in the back end. The controller would send dummy data to the front end at this stage. With a UI that would be able to receive data from the back end, the UI and the controller would be further modified so the UI would be able to send data to the back end.

With communication between the front and back end of the interview booker, new tables would be added to the database to hold the interview-invitation data. The controller would then be modified to communicate with the database, instead of using dummy data when communicating with the UI.

If the previous steps are successful and a functioning interview booker is integrated into the WA.works platform, the group will assess whether it would be feasible to extend the interview booker with a template feature.

3.1.2 Direct approach

In this approach, the required components are implemented directly in the existing WA.works front end application instead of creating an external React application.

This is the only difference between the dummy create-react-app approach and the direct approach. After implementing the components in the platform, the group proceeds as it would with the formerly explained approach.

3.1.3 Discussion of alternative approaches

With the dummy create-react-app approach, the group would have a blank canvas to work on and be able to focus solely on code pertaining to the interview booking part of the platform. As all the members of the group were inexperienced with Typescript and React, having a blank application to work with initially would reduce the risk of feeling overwhelmed and/or confused by Wide Assessment's existing code.

With the direct approach, the integration of the UI components would be reduced to simply exporting components and inserting them in the right place in the existing front end application. By initially implementing the UI components in an external React application, the group would have to modify or replace functioning UI components when integrating them into the WA.works platform. This could have turned out to be too difficult or too time-consuming.

After mapping out and discussing the two alternative approaches, the group consulted with Wide Assessment before making a decision. Wide Assessment recommended the group take the direct approach, pointing out that by working on the existing front end application, the group would have access to numerous components made specifically for the platform. The group decided to follow Wide Assessment's recommendation and proceed with the direct approach.

3.2 Specification

The group decided that the direct approach would be best. The main advantage was that the group could use the large library of React components already developed by Wide Assessment. In this way, the group could use premade components, already styled to match the rest of Wide Assessment's platform.

While it would be more overwhelming to start work in a labyrinthine folder structure, the group decided that it would be worth the reduction in time.

3.3 Selection of tools and programming languages

This subchapter contains a brief description of the technologies used in the solution of the project and a short discussion on why it was chosen where applicable.

3.3.1 Tools

Text editor: For code editing, Microsoft's Visual Studio Code was chosen. It was recommended to the group by Wide Assessment, and some of the group had prior experience with it. Visual Studio Code is a text editor with built-in debugging for programming and git commands for version control (Visual Studio Code, 2019).

Version control: Git was chosen as the version control system. The existing project's source code was given to the group by Wide Assessment through a private repository on github.com, which acts as remote storage for all project files.

Project management: To organize tasks and manage the groups' Scrum and Kanban board, the group used Zenkit. Zenkit is an application that allows groups to collaborate on calendars, to-do-lists, mindmap, and similar concepts, that allow better coordination of efforts (Zenkit, 2019).

Operating system: The group had initially planned on developing the project on the Windows 10 operating system, as the group was already familiar with this. To run the required docker software, the more advanced Windows 10 Education was needed and was provided by the school. However, after setting up and running the entire project for the first time, it was discovered that it suffered performance issues on the Windows operating system. It was then decided to switch to Ubuntu, which was the operating system used internally at Wide Assessment. In addition to the increased performance of the software, another advantage was that Wide Assessment could offer more assistance with Ubuntu.

Deployment: To avoid setting up all the required software involved in running a web service on every computer, the project used Docker. Docker is an operating-system-level virtualization program, meaning it can run a virtual image of another computer (Docker, 2019). This image contains all the required software and is simulated on each of the group's computers. Docker differs from a virtual machine, in that it does not simulate the entire operating system, but rather uses the host computer's operating system. This makes it more lightweight, and several images can be run simultaneously without impacting performance.

Communication: A communication service was needed, both with Wide Assessment and internally within the group. Slack was required for communication with Wide Assessment and, for convenience, it was also chosen as the main communication channel within the group. Slack lets users define text chat channels between certain members or wider channels with a specific theme (Slack, 2019).

Storage: For storage and collaboration on documents, Google Drive was chosen. Google Drive let the group work simultaneously on the same document while seeing other's changes in real time.

3.3.2 Programming languages

Front end: For front end development, Wide Assessment required the group to use typescript and the React library, as the existing front end was already written with these tools. Typescript is a superset of JavaScript which, in addition to normal JavaScript features, allow more advanced features like strict types and interfaces (Typescript, 2019). Typescript compiles to JavaScript, and runs client-side in the browser of the user.

React is a JavaScript library for creating user interfaces. React works by wrapping bits of HTML and JavaScript in components, making it easier to reuse and edit system-wide. Wide Assessment had already developed a huge amount of components based on the React library.

Back end: The project's back end was written in C# using Microsoft's .NET Core platform. .NET Core is a cross platform, open source re-implementation of the .NET Framework. It is actively maintained by Microsoft and a huge community of developers over on GitHub (Taylor, 2017). The group would mainly be working in the front end, but would need to add some functionality in the back end in order to facilitate communication with the database, where the interview data would be stored. C# is an object-oriented programming language (Microsoft, 2015), and the group was already familiar with this, or similar programming languages.

3.4 Project development method

3.4.1 Development method

For this project, the group chose an agile development method that encourages frequent inspection and adaptation. After seeking advice from Wide Assessment, who uses the Scrum method, the use of a simplified version of Scrum was decided.

Scrum is most commonly used in larger projects, and to implement the entire Scrum process felt unnecessary in a team made up of only three people.

Using the Scrum methodology means that the project is broken into smaller time restricted sections called sprints (Scrum, 2019). Lists of tasks are put together by the team to complete during each sprint of the project. When a sprint is complete, the progress will be tracked, and a new plan for the next sprint will be made. The goal of each sprint is to have a product that is incrementally improved from the last one. The group planned to have a short Scrum meeting each day discussing what was done the day before, activities for the current day and any new or ongoing issues.

For keeping track of tasks, the group uses the project managing platform Zenkit which provides a Kanban board on which you can place notes. These notes are tasks with different properties such as due dates and team members assigned to it. Two separate boards were created, one for the application development and one for the administrative tasks.

3.4.2 Project plan

See the attached Gantt diagram in Appendix 10.2.

3.4.3 Risk management

This section will discuss potential problems that might come up during the project. For a quick overview of risks and their severity, see appendix 10.1. The following is a more detailed description.

Lack of experience: There would be a chance that the group's inexperience with the technologies used and the already existing codebase will cause an inordinate amount of time to spend in the learning process. This might have lead to the group not having the time to complete the project.

The best way to avoid this would be a policy of not hesitating to ask for help when stuck in a problem, either asking other group members or the developers at Wide Assessment.

Poor time management: In a project of this scale, proper management of time is important. Too much or too little time spent on any one part of the project would have had a negative impact on the entire project.

To avoid this, the group would adhere to the Scrum method and have regular meetings to discuss allocation of time and resources.

The project cannot be used in the platform: If the finished interview invitation module did not comply with the specifications set by Wide Assessment, it could not have been used in their platform.

To prevent this outcome, the group would hold regular meetings with the developers at Wide Assessment to make sure they were headed in the right direction.

Poor communication within the group: Good communication within the group would be important. Bad communication could lead to duplicate work being done, or unnecessary bottlenecks.

To avoid this the group would be in constant contact on Slack and be careful to update the to-do and in-progress lists in Zenkit.

Poor communication with employer: To make sure the project stayed on track and did not stray from the specifications, good communication with Wide Assessment would be very important.

Fortunately, the group was offered to work at Wide Assessment's offices. This would allow the group immediate access to the developers, for questions and opinions.

School report interfering with work: As the group was tasked with both completing the project and writing a report on it, there would be a chance that the work on the report would take away resources the work on the project itself, and vice versa.

The group would have to decide in advance how much time to spend on each on a weekly basis.

Unexpected long term absence: There is always the risk of unexpected absence due to illness, accidents, etc.

There was little to do to prevent this, but the group decided that if this should come up, the person in question would make sure give notice as early as possible.

3.5 Evaluation method

Wide Assessment's intention when tasking the group with this project, was for the group to produce a usable component for their platform. In order for them to implement it into their existing frontend, the final result would have to meet all of Wide Assessment's specifications, in regard to both design and functionality.

The final evaluation would be whether Wide Assessment's developers felt the resulting component would fit well with the already existing platform. Having their final result integrated into the platform would be a success to the group.

4 SOLUTION DESIGN

This chapter provides a detailed description of the solution design.

4.1 Overview

The principal feature of the solution is being able to invite a candidate to an interview. This can be done in one of two ways:

- Inviting a single candidate through the chosen candidate’s page, or
- Inviting several candidates at the same time through the company’s list of potential candidates

In both cases, the interview invitation must be created in a modal, where the subject, content, location, etc. can be entered. In the modal, the date and time of the interview should also be entered, or in the case of inviting multiple candidates at once, dates and times for each individual candidate should be entered. Clicking the OK button must send the information to the backend, which will deal with the database and mailing.

A company user should also be able to access a page that lists upcoming and past interviews. This is a separate page linked to from the company’s list of candidates.

Both company users and candidate users should be able to see a page that details a single interview. A link to this page will be included in the email that is sent from the backend. This page should also include a button that adds the interview to a calendar through the AddEvent API.

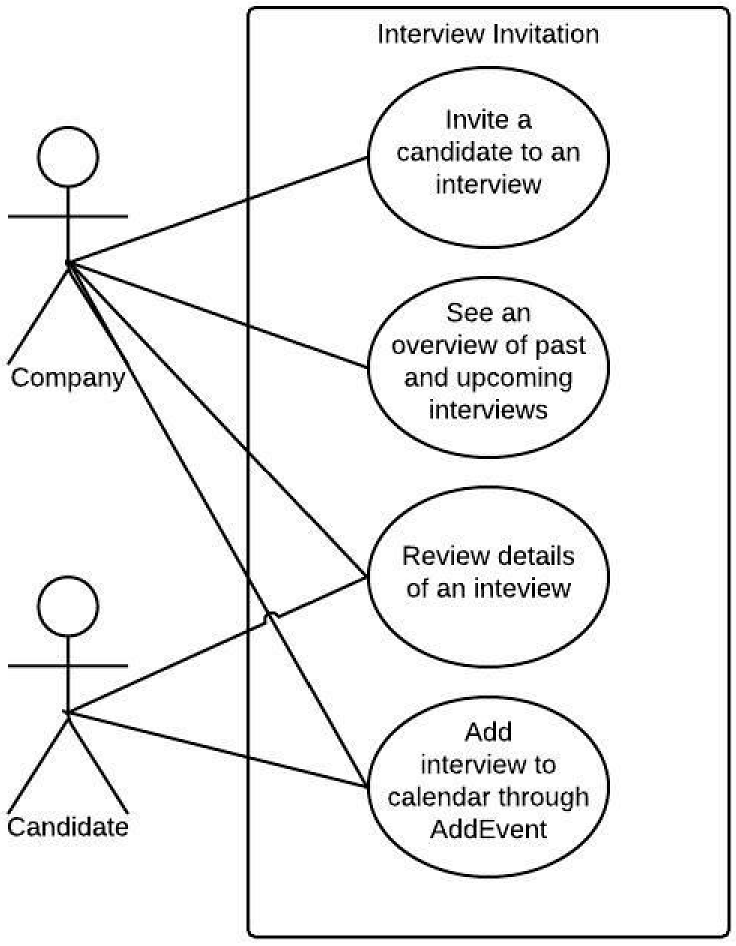

Interview Invitation
Invite a candidate to an interview
See an overview of past and upcoming interviews
Review details of an inteview
Add interview to calendar through AddEvent
Company
Candidate

Fig. 4.1 Use case diagram

4.2 Front end design

All of these features must comply with the graphical look and feel of Wide Assessment's existing platform. The existing modals on wa.works were examined, and the group came up with a basic design.

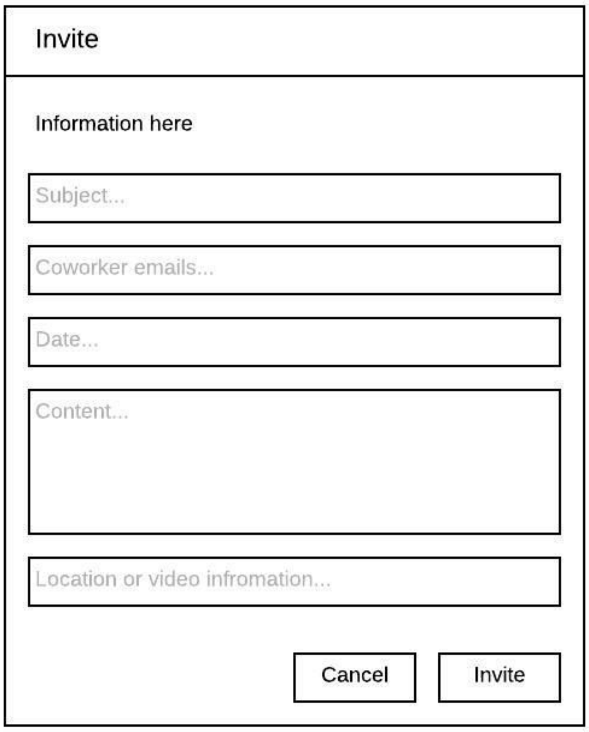
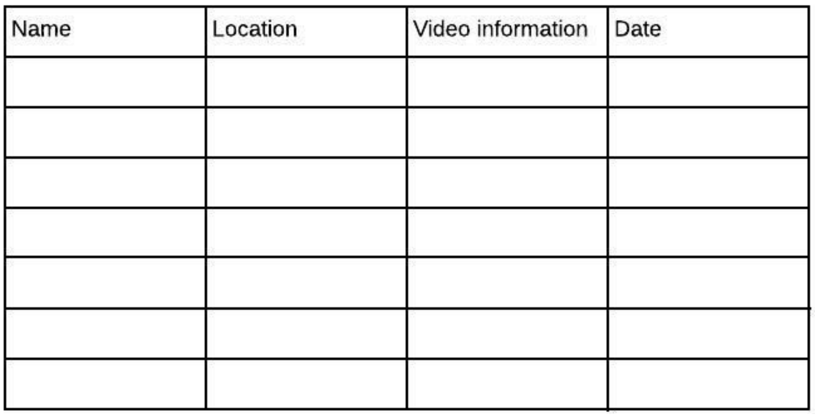


Fig 4.2.4 Original modal design

The specifications also required a list of sent interview invitations, visible to company users. It would have to display data like date and location. This would be contained in its own page, and not in a modal.



Name	Location	Video information	Date

Fig 4.2.5 Original interview list design

After designing the initial components, the next issue was integrating them into the existing platform. After discussions with Wide Assessment, it was decided to add an “Invite to interview” button on the page that displays a candidate’s profile.

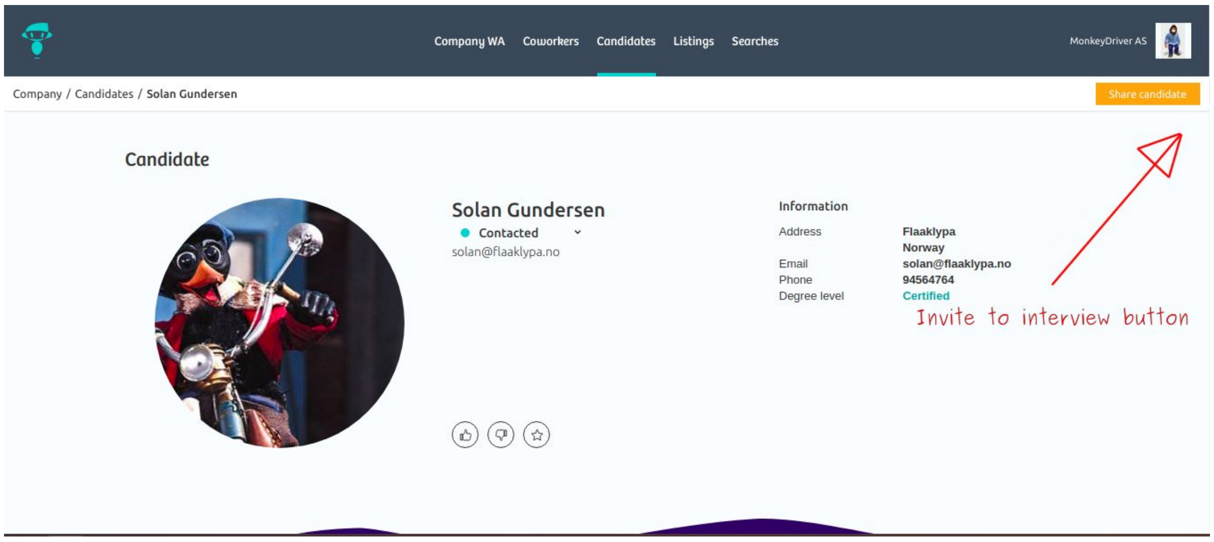


Fig. 4.2.1 Placement of "Invite to interview" button in the existing interface

The button to open the interview invitation list would be placed in a similar position in a company's list of candidates. An option to invite several candidates at once would be added to the bulk actions drop-down menu.

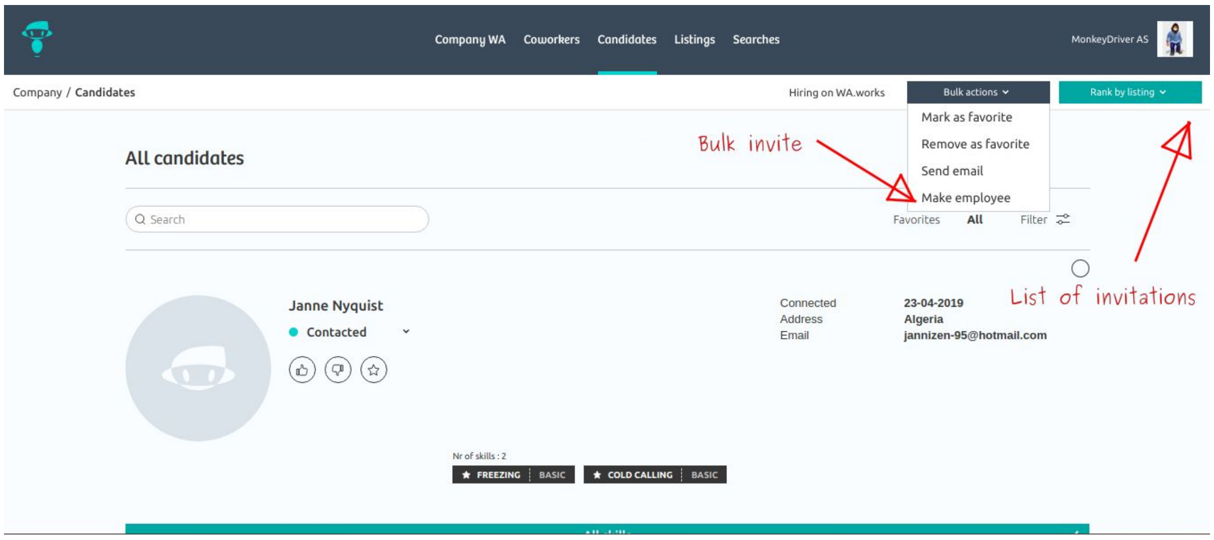


Fig. 4.2.3 Placement of bulk action and list of interview invitation buttons

4.3 Back end design

The solution needs to communicate with the database to store and load information about interview invitations. To accommodate this, the database must be updated with a new table.

In addition, a new controller must be created in the existing back end. The controller must be able to create new interview invitation objects and store them in the database, as well as update and delete existing entries.

Finally, the back end part of the solution should be able to send emails, to alert the involved parties that they have been invited to an interview.

4.4 Database table

The database needs a new table to store the data on the invitation interviews. The table must contain fields that record when, where and how the interview will be held, and information on who will be attending. A diagram was drawn using the Unified Modeling Language (UML) to specify the required fields of the new database table, and its relation to other tables in the database (see fig 4.4).

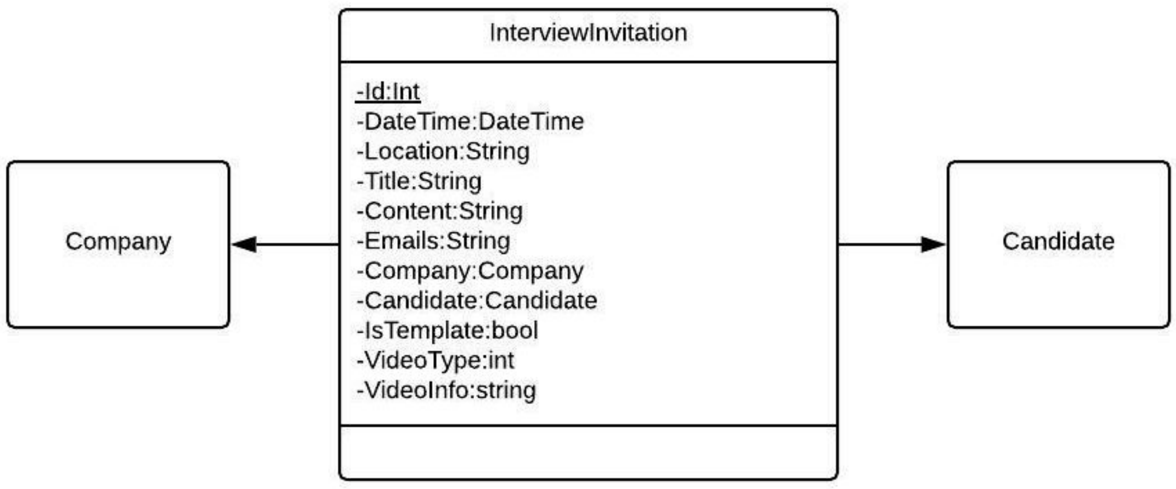


Fig 4.4 UML diagram

5 IMPLEMENTATION

This chapter describes how the solution was implemented.

5.1 How it works

This section will describe the use cases as implemented.

5.1.1 Sending a single invite

To invite a potential candidate, a company user navigates to the candidate's profile page and clicks the "Invite to interview" button. A modal opens where information about the interview is entered. When the user clicks "Invite" the interview invitation data is sent to a controller in the back end and stored in the database. Then the invitation is sent by email to the candidate.

5.1.2 Sending multiple invites

A company user can invite multiple candidates simultaneously by navigating to the "Candidates" page. The user marks the candidates to be invited and selects "Invite to interview" from the "Bulk actions" drop-down menu. A modal opens with additional date/time fields for each candidate. The interview invitation data for each invitation is sent to a controller in the back end and stored in the database. Then an invitation is sent by email to each candidate.

5.1.3 Add event to calendar

An URL is included in the invitation email which navigates to a page containing an AddEvent button widget. By clicking the button, the candidate can add the interview as an event to a calendar application.

5.1.4 Overview of invitations

A company user can get an overview of the invitations by clicking the "Interview invitations" button on the "Candidates" page. A new page is loaded containing a table with information about sent interview invitations. The information is retrieved via the controller in the back end from the database.

5.2 Front end

The wa.works platform is built on a framework called React. React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called "components" (React, 2019).

A component is a class or function that renders or returns a JSX object (a JavaScript-enabled pseudo-HTML), which is then compiled into HTML by React. It can also contain functions, such as validation logic or sending API requests. A component is usually a small piece of code, like a button or input field, but can also be larger and contain other components, like a modal or a component that contains the layout.

React also is a single-page framework. The entire app runs on a single HTML page and uses a process call routing to make the app look and feel like a multi-page app. If a user accesses the page "app.com/some/page", only the app at "app.com" is loaded, and the path, or route, "/some/page" is used be React to determine which part of the app to display.

When the project was started, a large library of components had already been written by Wide Assessment. The core feature of the project was to make a modal, and the existing library already contained a modal component. This component contained all the necessary CSS styling and functionality, like opening, closing and shading the backdrop. There also existed components to handle input, like text fields and checkboxes.

5.2.1 Making the interview invitation modal

The first step was to create a component that would contain the modal. The new component was specific to sending interview invitations and contained an instance of Wide Assessment's generic modal. The modal component was then filled with the appropriate input components, as well as button components for sending or canceling.

The group had initially planned to have the date of the interview input as a text string. The string would then be format checked and validated by a function in the interview invitation component, before being sent to the database API.

This turned out to be quite complicated, both to implement for the group and to understand for the user. Instead, it was decided to use a date-picker-style input for the date entry.

Wide Assessment did not have any premade date-picker components, and creating one from scratch was outside the scope of the project. Fortunately, React has a library of third-party components with an open license, and a suitable component called 'react-datepicker' was found.

A new component was made, similar to Wide Assessment's existing component for text input. Instead of a text input field, it contained the third party 'react-datepicker'.

First iteration

At the start of the project, the modal was made with the minimum input required to be functional. The initial implementation included fields for title, date, and location. It was opened by a button placed in an empty placeholder page, to make development simpler.

A function was implemented for validating the invitation. It would be called upon submission of the interview and would check for valid date and non-empty title and location fields. On an invalid interview, it would alert the user with a pop-up.

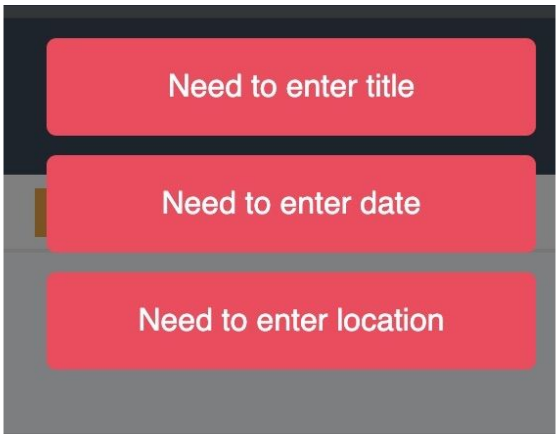
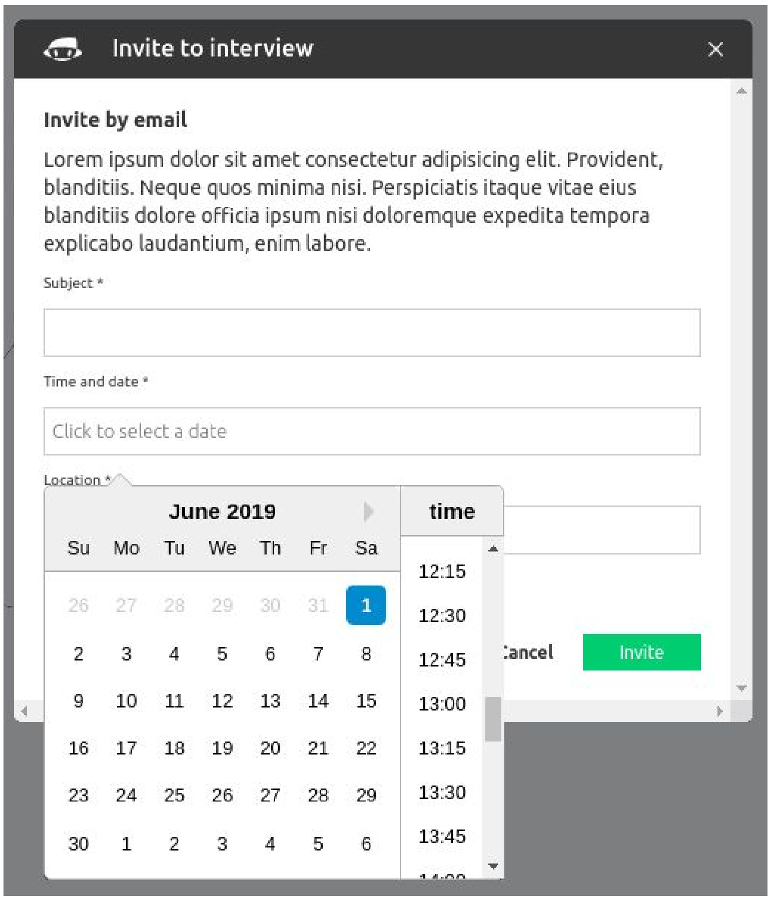

Need to enter title
Need to enter date
Need to enter location

Fig 5.2.1.1 Invalid data pop-ups

On submit with valid input, a POST request would be sent to the backend. Because the modal was placed in an empty page, it did not have access to the recipient candidate. In this iteration, a hard-coded id of a known candidate was sent instead.


Invite to interview

Invite by email

Lorem ipsum dolor sit amet consectetur adipisicing elit. Provident, blanditiis. Neque quos minima nisi. Perspiciatis itaque vitae eius blanditiis dolore officia ipsum nisi doloremque expedita tempora explicabo laudantium, enim labore.

Subject *

Time and date *

Click to select a date

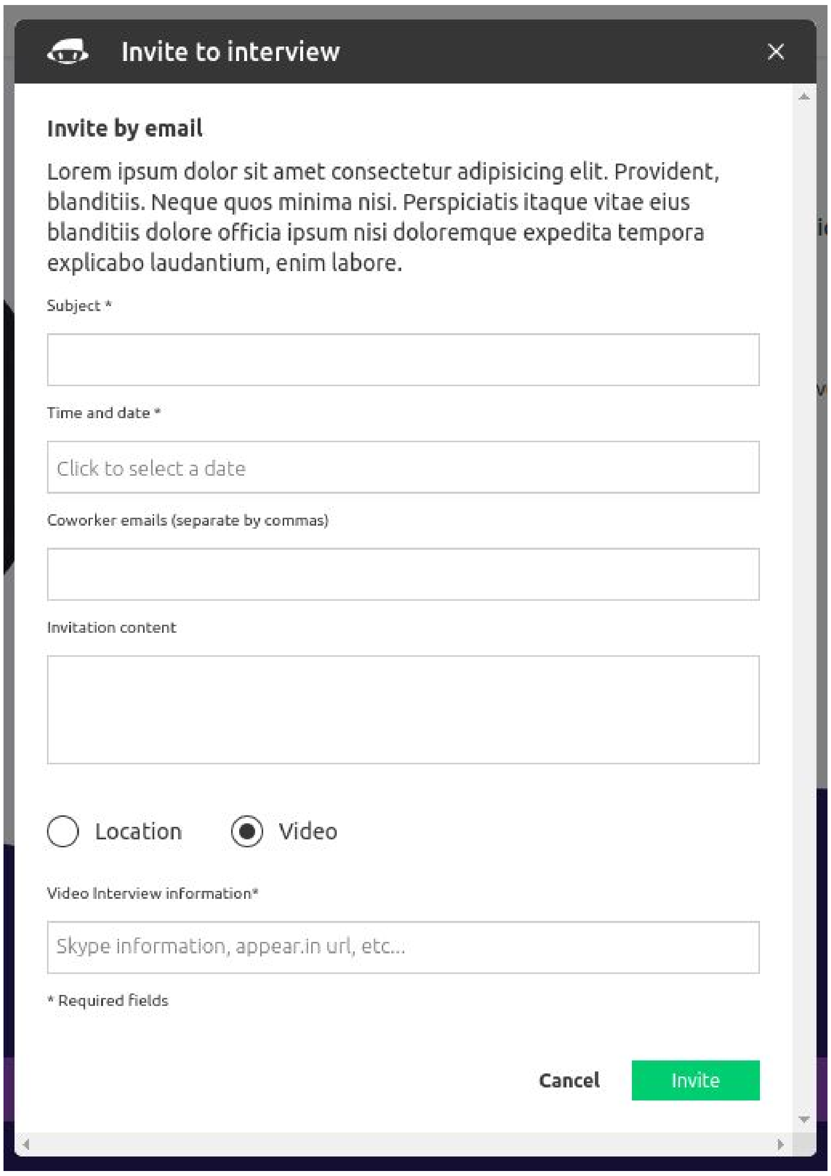
Location *

Fig 5.2.1.2 First iteration with basic input and date-picker

Second iteration

In the next iteration, the button to open the modal was placed on the page that displays a single candidate. In this way, the modal had access to the id of the currently displayed candidate.

Radio buttons were also added, to allow the company user to choose whether the interview would be held in person or on the internet in a video chat service.


Invite to interview

Invite by email

Lorem ipsum dolor sit amet consectetur adipisicing elit. Provident, blanditiis. Neque quos minima nisi. Perspiciatis itaque vitae eius blanditiis dolore officia ipsum nisi doloremque expedita tempora explicabo laudantium, enim labore.

Subject *

Time and date *

Click to select a date

Coworker emails (separate by commas)

Invitation content

Location [✓ Video]

Video Interview information*

Skype information, appear.in url, etc...

* Required fields

Cancel Invite

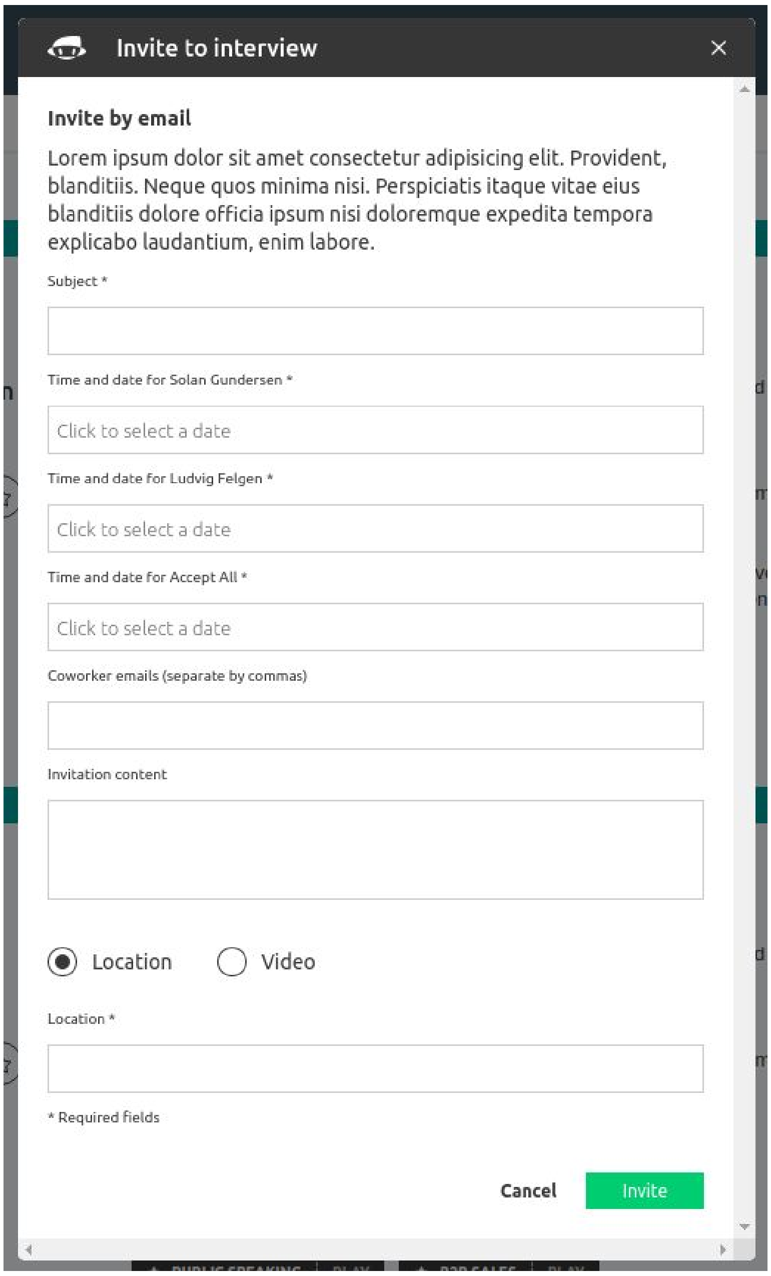
Fig 5.2.1.3 Second iteration

Third iteration

In the third iteration, interview invitations were added to the bulk actions menu on the page that displays all of a company’s candidates.

Instead of accepting a single candidate id, the modal was changed to accept an array of candidates. The modal would now display one date field per candidate, each labeled with the name of the candidate. The other fields were unchanged and other than the date, the same data would be sent to all candidates.

The single invitation was changed by taking an array with one id.


Invite to interview

Invite by email

Lorem ipsum dolor sit amet consectetur adipisicing elit. Provident, blanditiis. Neque quos minima nisi. Perspiciatis itaque vitae eius blanditiis dolore officia ipsum nisi doloremque expedita tempora explicabo laudantium, enim labore.

Subject *

Time and date for Solan Gundersen *
Click to select a date

Time and date for Ludvig Felgen *
Click to select a date

Time and date for Accept All *
Click to select a date

Coworker emails (separate by commas)

Invitation content

● Location ○ Video

Location *

* Required fields

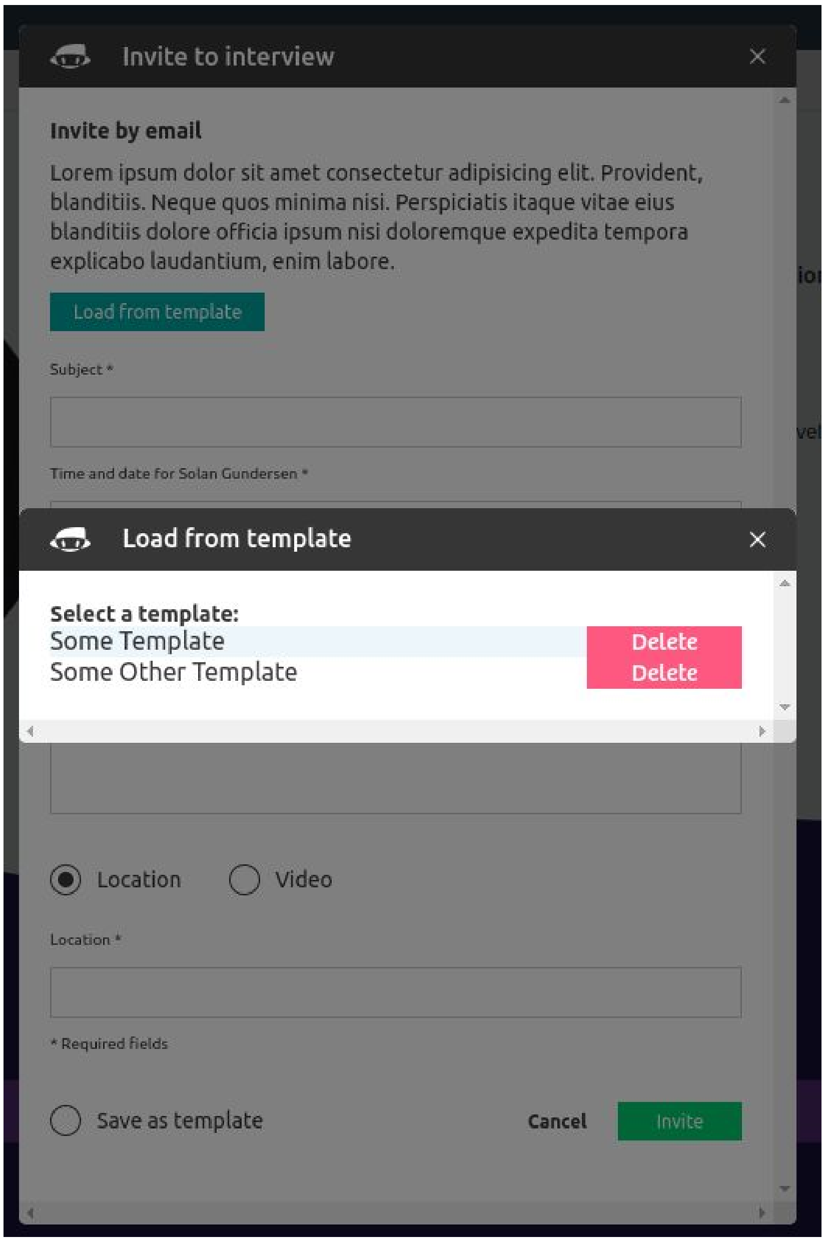
Fig 5.2.1.4 Third Iteratoin with bulk invite implemented

Final iteration

The last iteration added the feature of templates. A checkbox was added to the modal, allowing a company user to save the current interview invitation as a template. A template was stored as a normal interview invitation with its IsTemplate field set to true.

To load a template, a button was added to open another modal with a list of templates. The user would then select a template, and the title, content, emails and location/video info fields in the modal would be filled from the template.

The user was also able to delete templates in the template selection modal (setting the IsTemplate field to false).


Invite to interview

Invite by email

Lorem ipsum dolor sit amet consectetur adipisicing elit. Provident, blanditiis. Neque quos minima nisi. Perspiciatis itaque vitae eius blanditiis dolore officia ipsum nisi doloremque expedita tempora explicabo laudantium, enim labore.

Load from template

Subject *

Time and date for Solan Gundersen *

Load from template

Select a template:

Some Template	Delete
Some Other Template	Delete

(●) Location ( ) Video

Location *

* Required fields

( ) Save as template Cancel Invite

Fig 5.2.1.5 Final iteration with template function

5.2.2 Making the interview invitation list display

For the page displaying the overview of a company's sent interview invitations, it was decided it would contain two tables, one for past interviews and one for upcoming interviews. Wide Assessment's predefined table was used, in order to match the wa.works layout.

The Redux function “mapStateToProps” was used to get all interview invitations sent by the active company from the database as an array. The “componentDidUpdate” life cycle function was used to sort the initial array of interview invitations into two arrays, separating the upcoming interview invitations from the past interview invitations.

It was decided that when a company user lands on the interview invitations page, the table displaying past interviews would not be shown until a checkbox labeled “Show past interviews” was clicked.

5.2.3 Making the single invitation view

To be able to view data about an interview, a new page was made, showing relevant information. It was accessed by either clicking a link in the invitation list, or the link in the email sent to the candidate.

Users did not have to be logged in to view the page. A URL was generated from the specific interview’s GUID. A GUID is an acronym that stands for Globally Unique Identifier. Technically they are 128-bit unique reference numbers used in computing which are highly unlikely to repeat when generated despite there being no central GUID authority to ensure uniqueness (GUID, 2019). A GUID was necessary, because the normal ids were low numbers, and could easily have been found through trial and error by third parties. A GUID, on the other hand, was too long and complicated to be guessed.

The page also included a button to add the invitation to the user’s calendar. This was done through the AddEvent API.

The AddEvent API initially provided some problems. The AddEvent required the button to be rendered in the DOM from the first time the page was loaded, and not be replaced, but the page only rendered the button after checking whether the interview existed.

This was eventually solved by making React draw an empty button on the first page render, and after the interview was validated, fill in data inside the button instead of redrawing a new button with the information.

5.2.4 Managing the state

The wa.works platform uses Redux to manage the state. Redux is a predictable state container for JavaScript apps (Redux, 2015). Redux manages all calls to the backend.

The motivation behind using Redux is to keep all the other components from making asynchronous calls to the back end. This is to prevent unexpected changes to the application's global state. When all changes to the state are handled in one place, errors are avoided where the change in state is dependant on the current state.

To implement Redux, several files were created:

- A creators file, containing the actual REST request methods.
- An actions file, defining the Redux actions available, in this case, only setting the list of interview invitations in the state.
- A reducer file, linking the creator's get method with the state setting action.

5.3 Back end

The back end of wa.works is written in C#, using the .NET Core framework. The data is stored in a Postgresql database, connected to the .NET server with Entity Framework. The back end is an API built with ASP.NET MVC pattern.

5.3.1 The model

The database is written code first, so the first step in the back end part of the project was to create a class representing the interview invitation. The class needed to contain the following information:

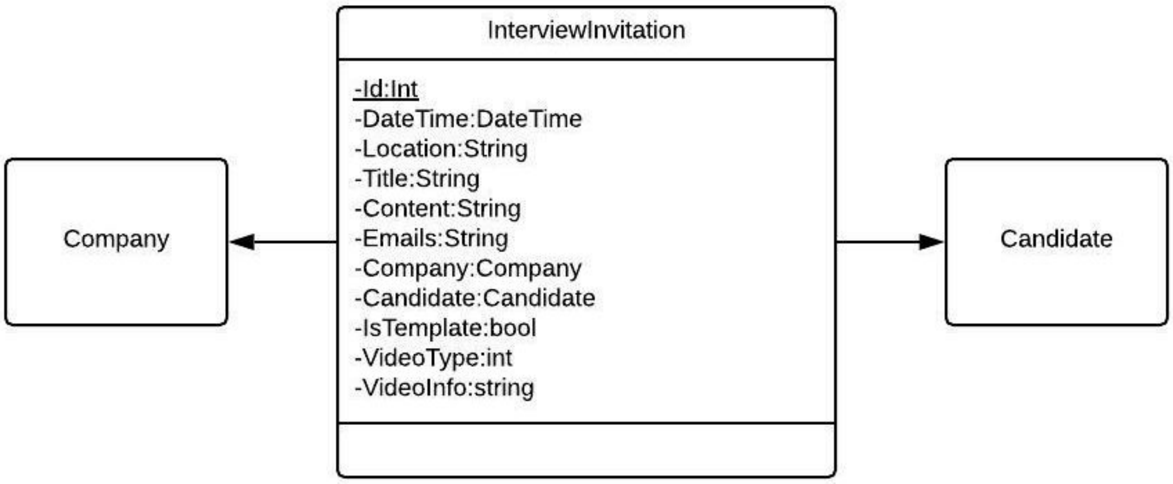
- Id: A primary key for the database
- Title: The header of the invitation, and the subject of the email sent to the invited parties
- Content: The text of the invitation, and in the email
- Email: A list of emails to send the invitation to, other than the invited candidate
- Date: The time and date of the interview
- Location: Where the interview will be held, if applicable
- VideoInfo: Info on how a video interview will be held, like a Skype username or appear.in link, if applicable
- VideoType: On what service a video interview will be held, if applicable
- IsTemplate: Whether or not the interview should be a template
- Company: The company that issued the invitation
- Candidate: The invited candidate

During the course of the project, some changes were made to the model.

A GUID (globally unique identifier) was added to serve as a link parameter when opening the single view page.

It was decided to include the video type in the video info string, deprecating the VideoType field.

The candidate object from the relational database is a connection between a user object and a company object. Since the candidate includes the company as a foreign object, the interview object's company field was superfluous.



5.3.2 Entity Framework

In order to create the actual table in the database, the class had to be migrated. Migration is a feature offered by Entity Framework that allows the user to add more tables, or add fields to existing tables, in an existing database (Microsoft, 2016). In this way, the database does not have to be dropped and recreated, and the data is retained.

With migration, the GUID could be effortlessly added to the database table, while retaining the testing dummy data in the database.

Removing columns, however, proved more difficult. The group ran into many errors when trying to migrate an updated class with removed fields. In the interest of saving time, it was decided to keep the video type and company fields, and not use them in the app.

5.3.3 Message

To facilitate communication between the front end and the back end, a message object was needed. A class was made, representing an interview invitation as it existed in the front end.

The message object contained the input data fields, but not id and GUID, as these would be assigned in the back end. It also did not contain a candidate object, just a candidate object id, so that the actual candidate could be found and attached in the controller.

When an HTML POST request was sent with a json formatted interview invitation, ASP.NET would translate the JSON object into the C# message object, so it could be used by the controller.

5.3.4 Controller

The controller was the main part of the back end. It contained the functions that could answer to the various REST requests, like GET, POST, PATCH and DELETE.

A GET method was implemented, to get a list of all interview invitations. A method for getting a single interview invitation was not needed, as redux would store all the interviews in the frontend. Finding a specific interview was then handled in the front end.

A DELETE method was also implemented, but there turned out to be no functionality in the frontend to delete interviews, so the method was unused.

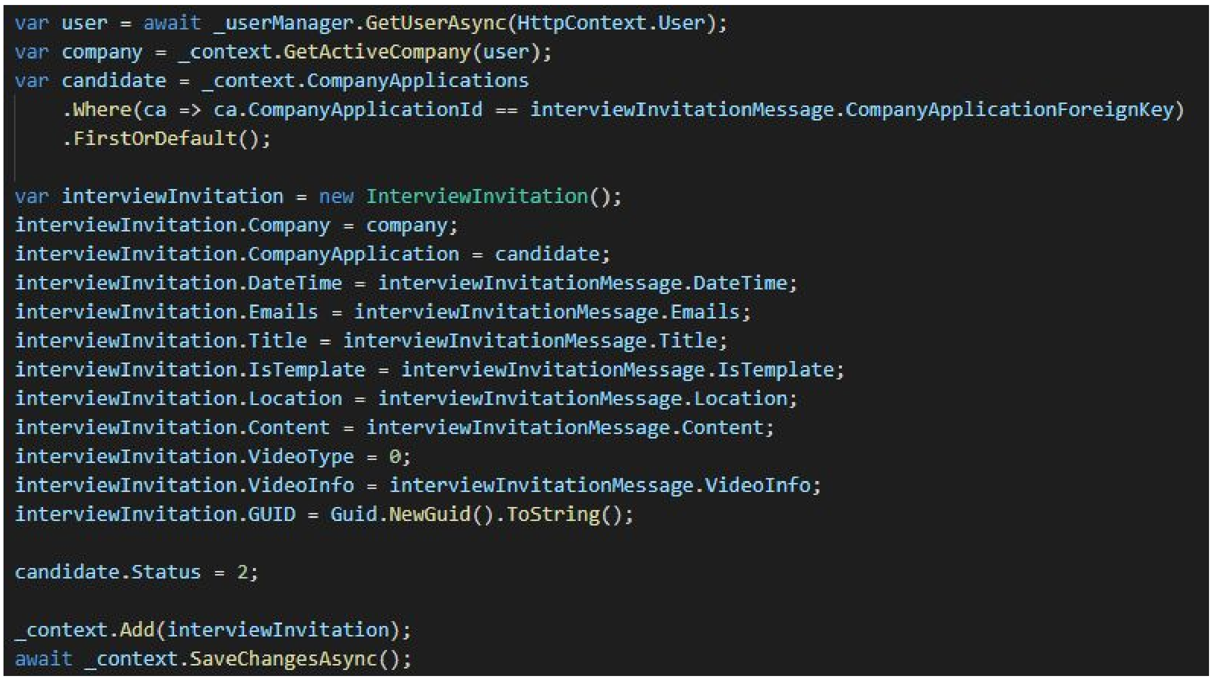
The PATCH method was made, to be able to disable a template. The would get an id as an argument, find the correct interview invitation, and set its IsTemplate value to false.

The POST method was the most involved of the REST methods. The POST method would be called when a company user clicked the Invite button in the front end, while all the data input fields were valid. It would receive an interview invitation message object as an argument.

The method would start by creating a new interview invitation model object, and the data fields would be copied from the message object. As the message only contained the candidate id, and not the candidate object itself, it had to be found in the database context. It was found using the LINQ methods Where and FirstOrDefault, with the id as the argument.

The new model object also needed a company object, but no data about the inviting company was included in the message. This was solved by using existing methods in Wide Assessment's backend, that returned the currently logged in user and the active company.

Finally, a new GUID was added with .NET's built-in NewGuid method, before the model object was saved to the back end.



```
var user = await _userManager.GetUserAsync(HttpContext.User);
var company = _context.GetActiveCompany(user);
var candidate = _context.CompanyApplications
    .Where(ca => ca.CompanyApplicationId == interviewInvitationMessage.CompanyApplicationForeignKey)
    .FirstOrDefault();

var interviewInvitation = new InterviewInvitation();
interviewInvitation.Company = company;
interviewInvitation.CompanyApplication = candidate;
interviewInvitation.DateTime = interviewInvitationMessage.DateTime;
interviewInvitation.Emails = interviewInvitationMessage.Emails;
interviewInvitation.Title = interviewInvitationMessage.Title;
interviewInvitation.IsTemplate = interviewInvitationMessage.IsTemplate;
interviewInvitation.Location = interviewInvitationMessage.Location;
interviewInvitation.Content = interviewInvitationMessage.Content;
interviewInvitation.VideoType = 0;
interviewInvitation.VideoInfo = interviewInvitationMessage.VideoInfo;
interviewInvitation.GUID = Guid.NewGuid().ToString();

candidate.Status = 2;

_context.Add(interviewInvitation);
await _context.SaveChangesAsync();
```

Fig 5.3.4 Part of the post method

Before saving the candidate's status would be set to 2. Status was an existing feature, and the value 2 would set a candidate's status to "Contacted" in the frontend.

After storing the interview invitation in the database, the post method would send emails to the candidate, and the included emails from the message, if applicable. The emails were sent by already existing methods, that would wrap the email content with the same header, footer, and style as other emails sent from Wide Assessment.

The contents of the emails were date and location, and the content text from the model. The emails also included links to the inviting company, and the single invite view, where the interview could be added to the user's calendar.

5.3.5 Response

The last part of the back end was the response object, which would be returned to the front end on a GET request. This would be constructed from the database model and sent to the front end as a JSON object.

The response contained all the data fields from the model, including the generated id and GUID. This would be converted to a JSON object by ASP.NET and sent to the front end.

6 EVALUATION

This chapter describes how the solution was evaluated.

6.1 Evaluation method

In addition to a final evaluation conducted by Wide Assessment, two methods of evaluation were chosen in order to continuously evaluate the solution and ensure the requirements were met throughout the development phase.

The first method of evaluation was functional testing of the application whenever a function was implemented or changed. Functions were tested by feeding them input and comparing the expected output with the actual output.

The second method of evaluation was letting the project owner evaluate the work. Weekly meetings were held with Wide Assessment's development team where the last week's work was reviewed. The meetings included a demonstration of the implemented functionality. This was to ensure the solution was developed according to the specifications of the project. The feedback from Wide Assessment was used to revise the course of the project development and make changes in the approach if necessary. In addition, a monthly meeting was held where the CEO attended to give feedback from a non-technical perspective.

6.2 Evaluation results

Functionally testing the solution continuously ensured that each part of the solution was implemented correctly according to the functional requirements. Functional testing exposed weaknesses and errors in the code early and decreased the risk of overlooking fatal flaws in solution.

The evaluations received from the meetings with Wide Assessment ensured that the solution complied with the company's specifications in regards to layout choices and programming standards. In addition, meeting with Wide Assessment frequently gave the group more confidence and motivation, which led to increased development efficiency.

When the solution was finalized, the group met with Wide Assessment for a final evaluation of the work, and a demonstration of the solution's functionality was shown. In addition, the developers at Wide Assessment had reviewed the code before the meeting. Wide Assessment was satisfied with the solution, regarded the project as a success and are planning to extend the wa.works platform with the Interview booker. The solution met the specifications given to the group, and the main goal to develop an interview-booking system in the wa.works platform has been reached.

7 DISCUSSION

Due to the lack of experience with the different technologies required for the project, it was decided the group would work from the office space offered by Wide Assessment. Working in close proximity Wide Assessment's own developers made it possible for the group to receive quick guidance and help when learning the new technologies in the pre-project phase, and throughout the development phase of the project. To further compensate for lack of experience, the group agreed to work all weekdays from 9:00 to 15:00.

The group was able to cooperate efficiently both internally and with the developers at Wide Assessment. Lack of experience was overcome, and the group was ahead of schedule enough to meet the initial requirements early in the development phase, making it possible to extend the solution with the multiple-invites and template function.

Using Zenkit and the simplified Scrum method made the project more manageable by splitting it into smaller tasks and providing a clear overview of the project's progress. Completing tasks gave a sense of progress, which in turn increased the group's motivation and efficiency.

The group has been ahead of schedule throughout the project, so it is clear that the time needed for development was overestimated. If the work could be done again, more time would have been allotted to learning React with Typescript and .NET Core in the pre-project phase, simply to increase the learning outcome for each member of the group.

8 CONCLUSIONS

The group’s judgment is that the goal of this project has been reached and the result is a solution with the following functionality:

- A company user can invite a candidate which he/she has a connection with, from the candidate’s profile page.
- A company user can invite multiple candidate’s which he/she has a connection with, with the “bulk action” drop-down menu on the “candidates” page
- A company user can see an overview of past and upcoming interviews, and add these to a calendar, through the “Interview Invitations” page, accessible from the “candidates” page
- An interview invitation can be saved as a template. Templates can be loaded to autofill all fields except the date field when sending an interview invitation.

Based on the results of the final evaluation, the group considers the project a success. The wa.works platform has been extended with an interview-booking system that is usable. It will be deployed as soon as a thorough revision of the group’s code has been done by the developers at Wide Assessment, and any necessary modifications or optimizations have been made.

9 LITERATURE/REFERENCES

Docker. (2019). Docs. Retrieved from
https://docs.docker.com/

Guid. (2019). What is a GUID? Retrieved from
http://guid.one/guid

Microsoft. (2019). Introduction to the C# Language and the .NET Framework.
Retrieved from https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

Microsoft. (2016). Code First Migrations. Retrieved from
https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/migrations/

React. (2019). Tutorial: Intro to React. Retrieved from
https://reactjs.org/tutorial/tutorial.html#what-is-react

Redux. (2015). Getting Started with Redux. Retrieved from
https://redux.js.org/introduction/getting-started

Scrum. (2019). What is Scrum? Retrieved from
https://www.scrum.org/resources/what-is-scrum

Slack. (2019). Features. Retrieved from
https://slack.com/intl/en-no/features

Typescript. (2019). Docs. Retrieved from
https://www.typescriptlang.org/docs/home.html

Visual Studio Code. (2019). Getting Started. Retrieved from
https://code.visualstudio.com/docs

Zenkit. (2019). Features. Retrieved from
https://zenkit.com/en/features/

Taylor, J. (2017, 19. juli). What is .NET Core. Retrieved from
https://cynicaldeveloper.com/blog/what-is-net-core/

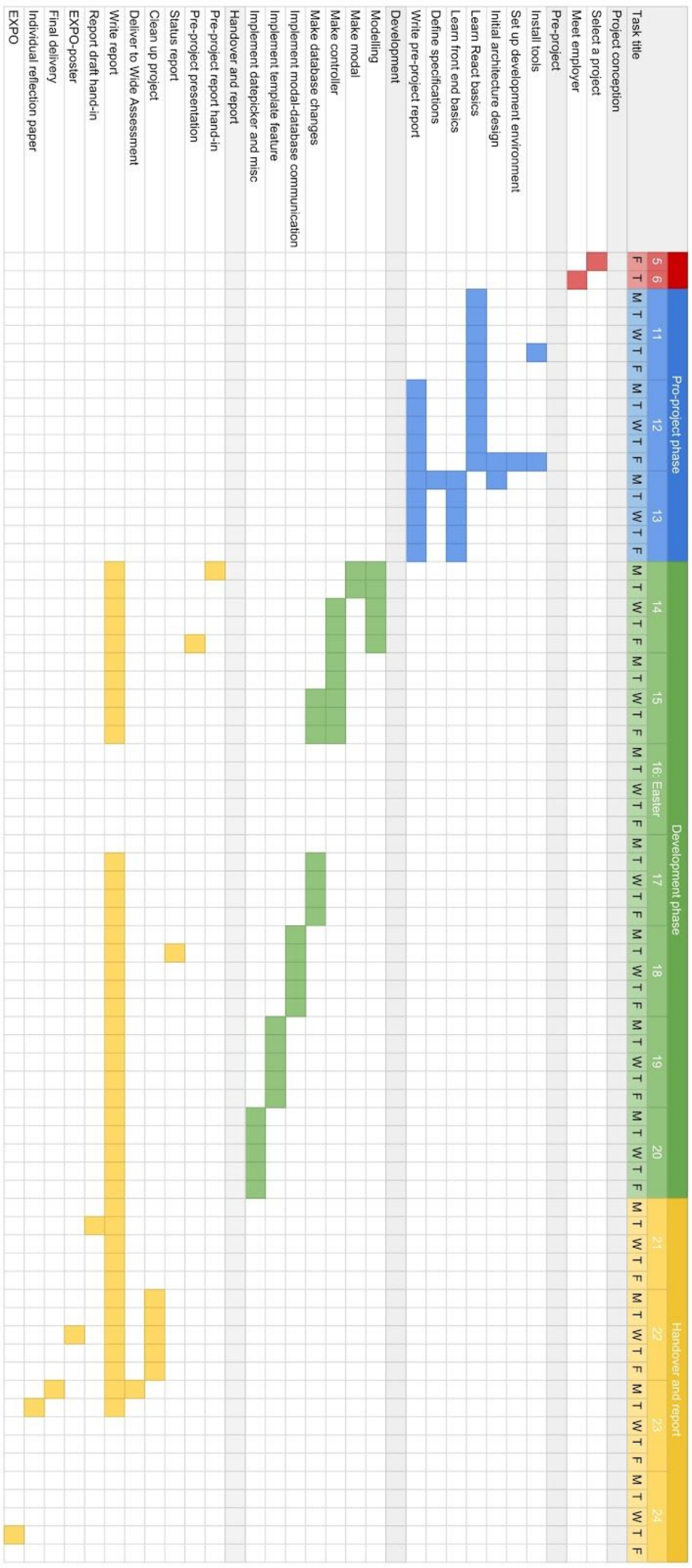
10 APPENDIX

10.1 Risk list

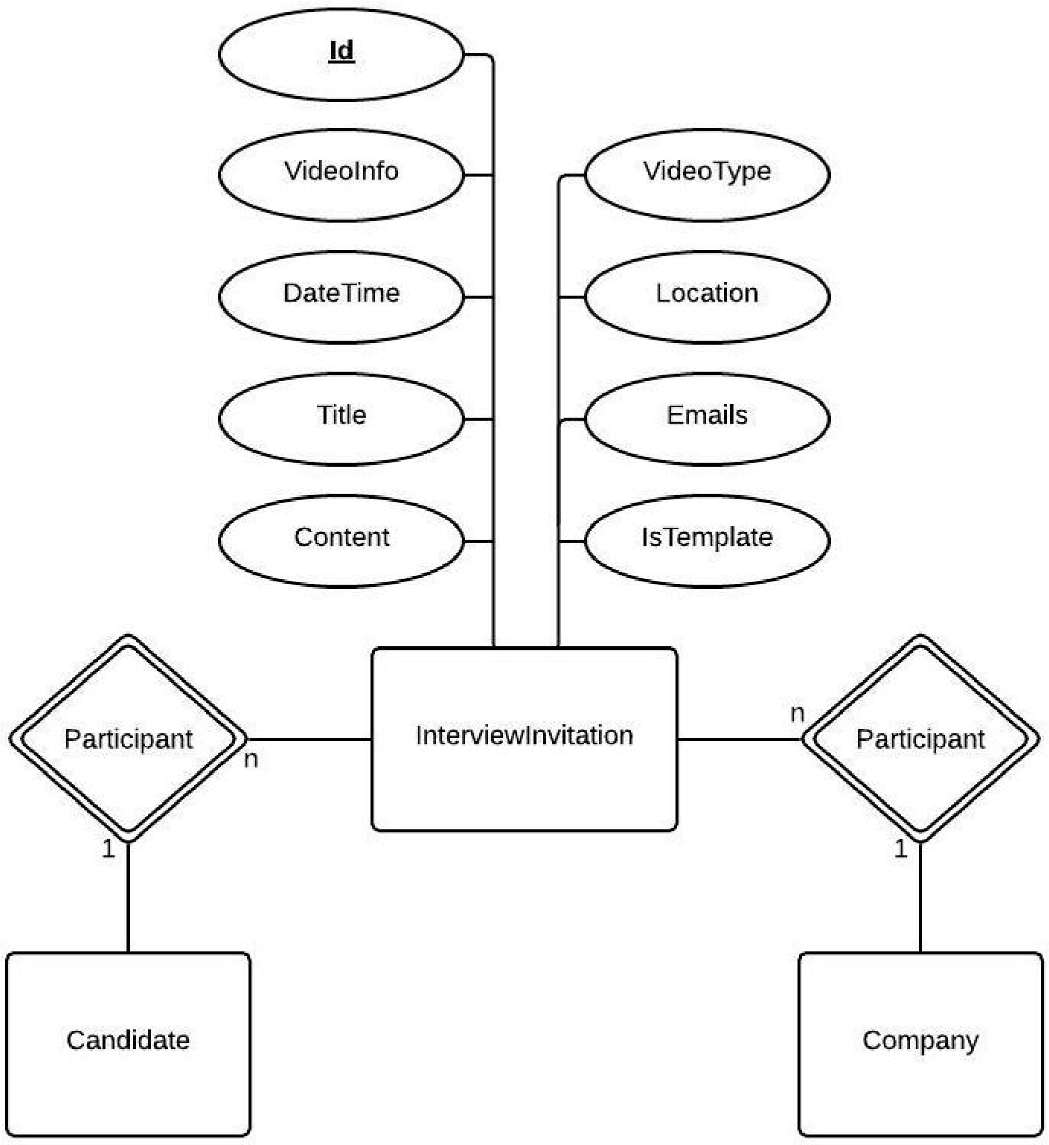
Problem	**Probabi lity**	**Impact**	**Risk factor**	**Measures**
Lack of experience	5	3	15	Seek help early
Poor time management	2	3	6	Delegate work, schedule time, occasional breaks, set deadlines
Not reaching WA’s specification, Project cannot be used in the platform	2	3	6	Agile development
Poor communication within the group	2	5	10	Patience
Poor communication with the employer	2	5	10	Don’t hesitate to call for meetings with the employer
School report interfering with work	2	5	10	Set time aside for report writing
Unexpected long term absence	1	4	4	Give early notice of absence

Table 10.1 Risk list

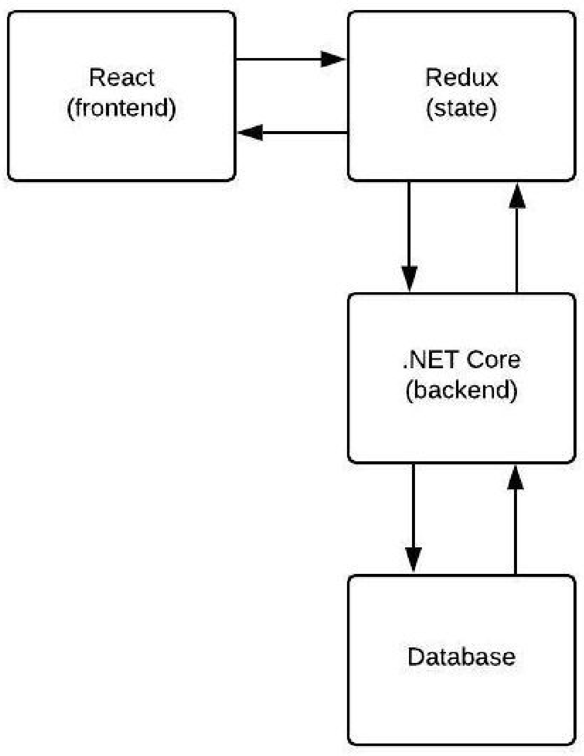
10.2 GANTT diagram



10.3 Diagrams

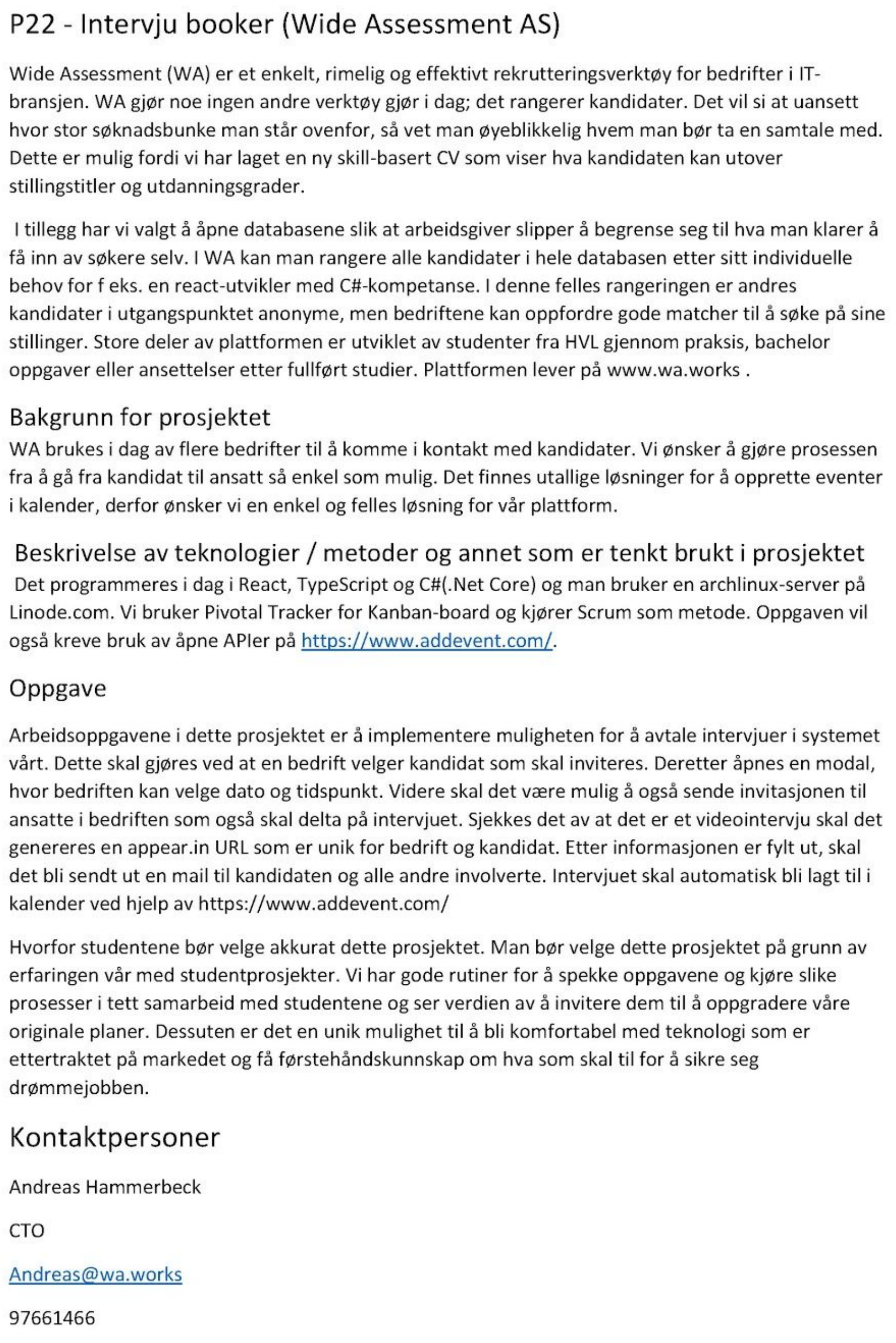


10.3.1 Entity relation diagram



10.3.2 Architecture diagram

10.4 Project description



P22 - Intervju booker (Wide Assessment AS)

Wide Assessment (WA) er et enkelt, rimelig og effektivt rekrutteringsverktøy for bedrifter i IT-bransjen. WA gjør noe ingen andre verktøy gjør i dag; det rangerer kandidater. Det vil si at uansett hvor stor søknadsbunke man står ovenfor, så vet man øyeblikkelig hvem man bør ta en samtale med. Dette er mulig fordi vi har laget en ny skill-basert CV som viser hva kandidaten kan utover stillingstitler og utdanningsgrader.

I tillegg har vi valgt å åpne databasene slik at arbeidsgiver slipper å begrense seg til hva man klarer å få inn av søkere selv. I WA kan man rangere alle kandidater i hele databasen etter sitt individuelle behov for f eks. en react-utvikler med C#-kompetanse. I denne felles rangeringen er andres kandidater i utgangspunktet anonyme, men bedriftene kan oppfordre gode matcher til å søke på sine stillinger. Store deler av plattformen er utviklet av studenter fra HVL gjennom praksis, bachelor oppgaver eller ansettelser etter fullført studier. Plattformen lever på www.wa.works .

Bakgrunn for prosjektet

WA brukes i dag av flere bedrifter til å komme i kontakt med kandidater. Vi ønsker å gjøre prosessen fra å gå fra kandidat til ansatt så enkel som mulig. Det finnes utallige løsninger for å opprette eventer i kalender, derfor ønsker vi en enkel og felles løsning for vår plattform.

Beskrivelse av teknologier / metoder og annet som er tenkt brukt i prosjektet

Det programmeres i dag i React, TypeScript og C#(.Net Core) og man bruker en archlinux-server på Linode.com. Vi bruker Pivotal Tracker for Kanban-board og kjører Scrum som metode. Oppgaven vil også kreve bruk av åpne APIer på https://www.addevent.com/.

Oppgave

Arbeidsoppgavene i dette prosjektet er å implementere muligheten for å avtale intervjuer i systemet vårt. Dette skal gjøres ved at en bedrift velger kandidat som skal inviteres. Deretter åpnes en modal, hvor bedriften kan velge dato og tidspunkt. Videre skal det være mulig å også sende invitasjonen til ansatte i bedriften som også skal delta på intervjuet. Sjekkes det av at det er et videointervju skal det genereres en appear.in URL som er unik for bedrift og kandidat. Etter informasjonen er fylt ut, skal det bli sendt ut en mail til kandidaten og alle andre involverte. Intervjuet skal automatisk bli lagt til i kalender ved hjelp av https://www.addevent.com/

Hvorfor studentene bør velge akkurat dette prosjektet. Man bør velge dette prosjektet på grunn av erfaringen vår med studentprosjekter. Vi har gode rutiner for å spekke oppgavene og kjøre slike prosesser i tett samarbeid med studentene og ser verdien av å invitere dem til å oppgradere våre originale planer. Dessuten er det en unik mulighet til å bli komfortabel med teknologi som er ettertraktet på markedet og få førstehåndskunnskap om hva som skal til for å sikre seg drømmejobben.

Kontaktpersoner

Andreas Hammerbeck

CTO

Andreas@wa.works

97661466