Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering: Data	Vår 6 semester, 2019 Åpen / Konfidensiell
Forfatter(e): Christoffer Holmesland Jørgen Melstveit Gaute Haugen	Signatur ....

Fagansvarlig: Erlend Tøssebro

Veileder(e): Erlend Tøssebro

Tittel på bacheloroppgaven: Webapplikasjon + mobilapplikasjon for bruk til interaktiv undervisning - DAT200

Engelsk tittel: Web application + mobile application for use in interactive teaching - DAT200

Studiepoeng: 20 stp

Emneord:	Sidetall: + vedlegg/annet: Stavanger, 15/05-2018

Forside for bacheloroppgaven
Det teknisk-naturvitenskapelige fakultet

Table of Contents

1 Introduction

2 Background

2.1 Babel

When making a web application with JavaScript it is important to remember that different browsers support a different range of the JavaScript features. Many new JavaScript features are not fully supported by all of the major browsers, e.g Google Chrome didn't have full ECMAScript 6 support for almost two years after the standard was defined. Some browsers also implement specific features which is only available in that browser. These features are often more performant and should preferably be used by the developer. To make it easier for the developer a tool called Babel can be used. Babel parses your JavaScript code and makes sure it can run on the specified browsers. If the code isn't supported by a given browser, the Babel tool will replace it with compatible code. In this project the Babel tool is part of the Vue build process. There isn't really any alternatives if the goal is to make your code browser compatible. Languages like CoffeeScript and Typescript will compile to JavaScript, but they don't have the same support in Vue as JavaScript does. Since the Vue build process by default includes Babel, this was chosen as the best option for this project.

2.2 Bootstrap

Since this web application is designed to work on both mobile and desktop devices, a way to easily place the different HTML elements depending on what kind of device the user has was needed. In web development this is called responsive design. Bootstrap is a web framework for creating responsive websites. It provides CSS classes which can be used to defined the layout of the page, and classes which can be used to define behavior of elements in a responsive environment. CSS for styling elements is also provided, this makes it easier for developers with little design experience to create a consistent style across the whole application. The project bootstrap-vue has been used because it contains predefined Vue components which are designed in a way that they interact well with the Vue reactive DOM. There are many alternatives to Bootstrap, two of them are Materialize.css and pure CSS. Materialize.css is very similar to Bootstrap, but it wasn't selected because it doesn't provide any clear advantages over Bootstrap, which the developers were already familiar with. Pure CSS wasn't picked because a lot of the functionality of Bootstrap would have been recreated, which didn't seem like a good use of development time.

2.3 Canvas

In web development there are three ways to display graphics to the user. HTML elements can be added and removed to the page using JavaScript, and elements can be styled using CSS. This is very slow because manipulating the DOM means that the whole page has to be rendered again. Another option is to use SVG (Scaleable Vector Graphics), which is done by adding elements to the SVG object. This is a good option, but it's not as performant as the Canvas where graphics are drawn using JavaScript. When visualizing graphs and other datastructures both SVG and Canvas are good options. However, since this project requires the user to interact with the datastructures in realtime, the canvas element was selected. The canvas element exposes a drawing API (Application programming interface), which lets the developer display things on it. Text, lines and other simple shapes can be drawn. The disadvantage of using canvas over SVG is scaling graphics. The user should be able to zoom in and out while modifying a graph. Compared to how this can be achieved with the canvas, the SVG solution is simple.

2.4 Connect History Api Fallback

The website in this project is built as a single page application, which means that there is only one HTML file, and extra content is loaded in using JavaScript when needed. Using the HTML 5 history object the user can be given the illusion that they are visiting other pages on the site. E.g the URL will show www.site.com/contact instead of www.site.com/index.html. If a user enters the URL www.site.com/contact in their browser, the web server will try to return a file to them called contact. This file doesn't exist and the user will get a 404 error. Connect-history-api-fallback is a Express middleware function which redirects the user request through the HTML file which results in the user seeing the page they expect.

2.5 Cookie Parser

HTTP doesn't store any state about who the user sending a request is. A problem resulting from this is that it's not possible to check whether the user is logged in or not. To solve this cookies can be included in the header field of the HTTP request. In Express you can access the cookies by parsing the header field of the request. To make this simpler for the developer, a package called cookie-parser can be used instead. cookie-parser is a middleware function which retrieves the cookies from the request and places them in the req object of the request handler.

2.6 Dotenv

When working on a application which talks to other systems, sensitive information about how to access the other systems are often needed. This can be database login information, or API access keys. This is information that shouldn't be shared with other people and it shouldn't be included in source control. The sensitive information is often stored in environment variables on the system running the application. In NodeJS, environment variables can be access from the process.env global object. The variables are passed into this object when the program starts. This is a good thing because then the values are not stored in the source code itself, however this also means that before running the application the variables has to be set every time. Instead the package Dotenv makes it possible to store these variables in a file. Dotenv will parse the file and load all the variables into the process.env object.

2.7 ESLint

When working with a langauge like JavaScript which doesn't check for errors before you run the code, it can be useful to have a tool to detect errors before the program is ran. ESLint is a linter for JavaScript. A linter is a tool which analyzes code and warns the programmer about errors. When working in a team with several people the code can often get harder to read over time if the developers use different coding styles. To prevent this, it's possible to configure ESLint to also check coding style. This project is configured to use the Prettier option which means that coding style will be enforced.

2.8 ExpressJS

If you want to create a web application, users need somewhere to get the content from. This is where a web server is used. A simple server which only returns a HTML file is fine if the application only contains static files, but if it's more complex, a web framework should be used. ExpressJS is a web framework for NodeJS. By default it doesn't come with a lot of feature. It's possible to define handlers for different URLs, and it can start a HTTP server, but it's strength is middleware functions. Middleware functions are functions which does something with either the request or response objects before the actual URL route handler function sees them. This design means that Express is very modular and it's only needed to include what will actually be used. This is why Express was chosen over the simple HTTP server which is available in NodeJS. Instead of Express, Meteor could have been used. Meteor was not chosen because it works best with websites where a lot of the content is static, which isn't true for this application. It also lacks good support for SQL databases which is something this application uses.

2.9 NodeJS

JavaScript has traditionally been a language which could only be used in web browsers. This means that client code had to be in JavaScript and server code had to be in another language. When working on the same project it is often easier to only use one langauge. NodeJS is a JavaScript runtime which makes it possible to run JavaScript code on both client and server side. Because the code doesn't run in a browser, it's possible to access the file system and the operating system of the machine running the application. NodeJS comes with a command line tool called npm (node package manager) which can be used to install packages in your project. Packages are code which other developers have written and shared. This feature makes it easier to not "reinvent the wheel" when creating your application. There is only one viable alternative to NodeJS, which is Deno. Deno is made by the same person as NodeJs, but is supposed to be more secure. One of ways of achieving this is by removing the npm tool, which is the reason why NodeJS was chosen over Deno.

2.10 PassportJS

PassportJS provides a set of middleware functions for authenticating users using OAuth. It uses strategies to correctly authenticate users. It's used together with the passport-openid-conncet package to let users use their Feide account on the Interaktiv Undervisning site.

2.10.1 OAuth 2.0

OAuth 2.0 is the most used authorization protocol today. The OAuth protocol was originally developed for systems to easily and securely get access to user information stored on other systems over the internet.
OAuth 2.0 work flow consists of the client, that is the third party system that wants information about the user, first redirects the user to the authorization server together with wanted scope, in order to authorize the user. After the user has been authorized, the user will be asked whether or not he will consent to allow the client access to data requested. What kind of data and what the client can do with them is determined by the type of scope that is sent with the request. If the user consent the user will be redirected back to the client page on a callback url and an authorization code is given to the client. This code is later sent back to the authorization server in order to authorize the access to the user's resource owner. If authorization code is valid an access token is given to the client which the client can use it to access the users resource server and obtain the requested data.
Most of the webpage redirects happens on front channels, while obtaining and using the authorization token happens on the back channel. This is done in order to make sure that no sensitive data is accidentally intercepted by someone with malicious intent. It also guarantees that even if someone managed to intercept the authorization code sent from the authorization server ahead of time, they still wouldn't be able to use it on the user's resource server. This is because the resource server requires a high secure tunnel on the server side.

2.10.2 OpenID Connect

The OAuth protocol was never designed for authentication, it was only designed for authorization and permissions over the web. OpenID Connect is a layer that was placed on top of the

OAuth 2.0 protocol in order for OAuth to also be able to handle authentication.
OpenID connect allows OAuth 2.0 to store an ID token, which will be used to get information about the user. With this extra user information OAuth 2.0 can be safely be used as an authentication protocol. In this project a combination of OAuth 2.0 and OpenID Connect was used in order for the web application to be able to authenticate students through uninett's dataporten.

2.11 Socket.IO

Socket.IO is the most commonly used WebSocket for Node.JS, which makes it the most supported and with the most active community of the alternatives. Socket.IO is built using the Engine.IO and creates a connection between the server and the client in real time, with bidirectional event-based communication. In this application, multiple users will send updates to the server, and the server will send out various updates to numerous clients in real time. This would have been very complicated and time-consuming to get right with AJAX or regular HTTP requests. When a Socket.IO function is set up, it's easy for the functions only to send the information that is meant for that client. Socket.IO also have a feature called "rooms" where the server can add a connection to a virtual room. Then when a quiz is running, and the server needs only to update clients connected to that quiz. It will broadcast it's messages to that room, and then only clients connected to that room will get that message. This makes Socket.IO ideal for making the server handling more than one quiz at the time. Also, all the socket.IO functions are asynchronous which makes the wait time for a response a lot shorter.

2.12 SQLite3

SQLite3 is a SQL database engine which doesn't require a separate server process to operate. Instead SQLite3 reads and stores all the data directly to a single stored file. This is one of reasons why SQLite databases are used in a lot of small and medium sized web pages. This is primarily the reason why SQLite was used in this project over other database engines.
By default if you try to connect to a SQLite3 database which doesn't exist it will automatically create a new database file for the requested database. SQLite3 has support for just a few data types such as INTEGER, TEXT and BLOB. However, it does support all SQL features, which makes it both easy and effective to use.

2.13 Vue

Vue is a framework in Node.JS for building user interfaces. Vue makes a single page application which makes the load in one HTML file and the javascript and CSS needed for the first page. Then when the user clicks around on the page the main javascript loads in component or views in as more javascript and css. Then the DOM gets updated with the new components. These updates make the navigation and interaction on a webpage feel more smooth and responsive. When programming the client side of the application, the HTML, CSS, and javascript will get its own container inside the Vue file. When you have written the application one needs to use the Vue CLI to compile the code.

2.13.1 Vue Cli

The Vue CLI is used as a developer tool to start, build, manage and deploy a Vue project. When setting up a Vue project, there are some options to how you want to set it up and what plugins you want to use. Some of the plugins available help with linting, templating the "pages" and setting up Vue features. One of those features that was used in this project is Vue router.

2.13.2 Vue Router

Vue router tries to mimic the multipage application by using routes that are URL paths and views that are Vue component that should be used to be the template for a page. This leaves the user with the impression that the single page app acts like a multipage app, but the website feels more smooth and responsive as the webpage don't load in another HTML file. It replaces elements in the DOM using javascript.

3 References

4 Attachments