My Best Software Technology Evaluation Project Ever

Jonatan Valen

November 23, 2023

Abstract

10-15 lines with the software technology and the highlights from the project that has been undertaken.

1 Introduction

1.1 Development of the Feedback App

In this project we have developed a feedback app for voting on polls. We have implemented register and login functionality. A user can create polls that are either private or public, and both users and voters can vote on polls. Only logged in users can vote on polls. We have also created a pollinfo display where users can look at results of their own polls. On this pollinfo page, you can also find an shareable link so that people that dont have user accounts can vote on public polls. Users can also manage their polls by ending them, starting them again, and deleting them.

1.2 Technology stack

The technology stack chosen for this project is both robust and dynamic, encompassing a spectrum of modern development tools and frameworks. We have employed React for our front-end development, capitalizing on its component-based architecture for a responsive and interactive user interface. The back-end is powered by Spring Boot, providing a strong and scalable foundation for our application services. For message handling and asynchronous operations, RabbitMQ has been integrated, ensuring efficient communication and process management.

Further enhancing our application's capabilities, we have incorporated Tailwind CSS for streamlined styling, JPA (Java Persistence API) for efficient data handling, and Hibernate for object-relational mapping, ensuring a seamless data management experience. Firebase is used for our authentication solutions, fortifying the app's security. Additionally, Dweet.io has been utilized for its simplicity in real-time messaging, and an H2 database for its lightweight and easy-to-use nature.

1.3 Results

During this project, we have gained a better insight into different software technologies, frameworks and tools. We now have a working prototype of a voting app which can easily be extendable to include more features in the future.

1.4 Overview

This rest of this report is organised as follows: Section 2 gives an overview of the software technology stack used. Section 3 describes the design of the prototype. Section 4 details how the prototype has been implemented. Section 5 explains how the prototype has been tested and what experiments have been done. In Section 6 we provide conclusions regarding different aspects of the project.

2 Software Technology Stack

2.1 React and Tailwind: Frontend Infrastructure

For our frontend, we chose to use React and Tailwind. React, developed and maintained by Facebook, is a declarative, efficient, and flexible JavaScript library for building user interfaces. It allows us to create large web applications that can change data, without reloading the page. Tailwind CSS is a utility-first CSS framework that has been gaining popularity for its unique approach to styling web applications. Here's a detailed explanation of why React and Tailwind CSS is an excellent choice for the front-end development of our web application:

2.1.1 Commponent-Based Architecture

React's core strength lies in its component-based architecture. This approach enables developers to build encapsulated components that manage their own state, and then compose them to make complex user interfaces. By breaking down the UI into individual components, React promotes reusability, which can significantly speed up development time and reduce code redundancy.

Each component has a well-defined lifecycle, and React provides hooks into these lifecycle events. For instance, developers can execute code at the moment a component mounts, updates, or unmounts, giving fine-grained control over its behavior.

2.1.2 Declarative Nature

React is declarative, meaning that developers write code that describes the UI, and React takes care of updating the Document Object Model (DOM) to match that description. When the state of a component changes, React updates only the parts of the DOM that have changed. This declarative update

process simplifies the code and makes it more predictable, which is crucial for the maintainability of large applications.

2.1.3 Strong Community and Ecosystem

React has a large and active community, which translates into a wealth of third-party components, libraries, and tools that extend its capabilities. The ecosystem around React includes state management libraries like Redux and MobX, routing solutions like React Router, and numerous other utilities that complement React's feature set.

2.1.4 Developer Experience

Developer experience is another strong suit for React. It offers a robust set of developer tools, including React Developer Tools, which is a browser extension for debugging component trees. It also provides meaningful warnings and error messages that help developers quickly identify and resolve issues.

Tailwind CSS enhances developer experience by allowing them to stay in the HTML for the majority of the styling work, which reduces context switching between HTML and CSS files. This can lead to faster development cycles and increased productivity. The utility classes are also self-explanatory, which makes the code easier to read and understand at a glance.

2.1.5 Utility-First Approach

Tailwind CSS employs a utility-first approach to styling, which is a significant shift from traditional CSS frameworks that provide UI kits with predefined component styles. Instead of using semantic class names tied to specific components, Tailwind provides low-level utility classes that can be composed to build any design, directly in the markup. This approach makes it incredibly flexible and reduces the need to write custom CSS to an absolute minimum.

2.1.6 Performance

Because Tailwind encourages the use of utility classes, the size of CSS files tends to be smaller, which can lead to performance gains. When combined with its JIT compiler, Tailwind generates the minimal amount of CSS needed for a project, leading to faster load times and improved overall performance of the web application.

2.2 Spring Boot: Back-End Infrastructure

Spring Boot is an extension of the Spring framework that simplifies the initial setup and development of new Spring applications. It favors convention over configuration and is designed to get you up and running as quickly as possible. Here's a detailed explanation of why Spring Boot is an excellent choice for our back-end infrastructure:

2.2.1 Simplified Configurations

Spring Boot simplifies the management of application configurations. It eliminates the need for defining boilerplate configuration with its auto-configuration feature. This feature automatically configures your application based on the libraries present on the classpath. This can significantly reduce development time and increase productivity because developers can focus on the unique aspects of their application rather than on infrastructure setup.

2.2.2 Standalone Applications

Spring Boot makes it easy to create standalone, production-grade Spring-based applications that can be run directly from the command line without requiring an external web server. This is made possible by embedding servers like Tomcat, Jetty, or Undertow directly in the application. The ability to run as a standalone app simplifies both the development and the deployment processes.

2.2.3 Opinionated 'Starter' Dependencies

To streamline the configuration process, Spring Boot provides a set of 'starter' dependencies that bundle the necessary libraries to get a feature working. These starters cover many Spring Boot features, such as data access, messaging, and web services. Using starters, you can avoid library version conflicts and ensure that you're using a set of dependencies that Spring Boot has tested and approved.

2.2.4 Community and Support

Spring Boot benefits from the strong support of the vibrant and extensive Spring community. There are numerous guides, tutorials, and a comprehensive reference documentation available. Moreover, the community support forums and Stack Overflow are active and helpful in solving problems.

2.2.5 Database Integration and ORM Support

Spring Boot has excellent integration with Spring Data, which simplifies database access and provides support for various data access technologies, including JPA, JDBC, and NoSQL. Spring Boot also

works well with Hibernate - one of the most popular ORM tools - making it easier to work with databases using an object-oriented paradigm.

2.3 RabbitMQ and Dweet.io: Message Systems

RabbitMQ is an open-source message broker renowned for its reliability, scalability, and flexibility, making it an exceptional choice for complex application messaging systems. It allows applications to communicate asynchronously through a platform-neutral, broker-based system. The decision to use RabbitMQ for message handling within the application is based on several compelling features:

2.3.1 Core Functionality and Message Queueing

RabbitMQ acts as an intermediary, handling the exchange of messages between services. It offers support for various messaging protocols, intricate message queuing, delivery acknowledgment, and multiple exchange types, ensuring flexible routing to queues.

2.3.2 Asynchronous Processing

With its ability to facilitate asynchronous messaging, RabbitMQ enables the queuing of messages to be processed by consumer services at their own pace. This separation of producers and consumers allows each to scale independently, enhancing system robustness and reliability.

2.3.3 Reliability and Durability

RabbitMQ ensures the durability and persistence of messages, safeguarding against losses during system failures. Its persistent messages and durable queues allow for system recovery without data loss, which is vital for maintaining application integrity.

2.3.4 Scalability

Designed for high-load environments, RabbitMQ can be extended across clusters and federated nodes, managing a large volume of messages and multiple client connections. Features like consistent hashing and sharding facilitate efficient load distribution.

Complementing RabbitMQ, Dweet.io provides a streamlined approach to real-time messaging, particularly advantageous for Internet of Things (IoT) applications. Dweet.io simplifies the real-time broadcasting of messages from devices, allowing for seamless data subscription:

2.3.5 Simplicity and Real-Time Communication

Dweet.io's straightforward design enables devices to "dweet" messages, which can be instantly accessed by clients or applications subscribed to that feed. This real-time capability is crucial for scenarios where timely data delivery is of the essence.

2.3.6 Ease of Integration

Dweet.io's non-complex setup makes it an accessible tool for real-time messaging. It requires no preliminary configuration, allowing for immediate deployment and integration into existing systems.

In combination, RabbitMQ's robust message handling and Dweet.io's real-time messaging capabilities provide a comprehensive solution for the application's communication needs. RabbitMQ manages the heavy-duty, reliable queuing and processing of messages, while Dweet.io offers the agility and ease required for real-time data dissemination, ensuring a responsive and interconnected application ecosystem.

2.4 JPA, Hibernate and H2: Data Management and Storing

2.4.1 Java Persistence API (JPA)

Java Persistence API (JPA) is a Java specification providing an Object-Relational Mapping (ORM) standard. This abstraction simplifies the interaction between Java objects and relational databases. JPA's primary advantage lies in its database-agnostic nature, offering flexibility in database choice and minimal application code changes when switching databases.

2.4.2 Hibernate

Hibernate, as an implementation of JPA, enhances its capabilities. It's renowned for its performance optimization features, such as lazy loading and sophisticated caching mechanisms. Hibernate also supports a rich set of querying capabilities through the Hibernate Query Language (HQL) and Criteria API, enabling more complex and efficient data retrieval operations. Together, JPA and Hibernate offer:

- **Simplified Database Interactions**: By abstracting complex JDBC operations, they reduce boilerplate code and streamline database interactions.
- **Advanced Query Capabilities**: Enhanced querying and retrieval options facilitate sophisticated data handling.
- **Performance Optimizations**: Features like caching and batching improve application performance and database interaction efficiency.

2.4.3 H2 Database: Efficient and Lightweight Data Storage

The H2 database is an in-memory, Java-based database known for its speed and simplicity. Primarily used in development and testing environments, H2 stands out for its ease of setup and rapid execution. It offers a web console for direct database interactions, enhancing its usability during development.

2.5 Firebase: Authentication (New Technology)

Firebase Authentication provides a full backend service that can authenticate users through multiple methods, including passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. The decision to use Firebase for authentication in the application is based on its comprehensive suite of features that enhance user security and improve the overall user experience.

2.5.1 Seamless User Authentication

Firebase Authentication offers a seamless integration with applications, providing a complete identity solution supporting email/password auth, social media login, and phone authentication. Its SDKs and ready-made UI libraries allow for quick implementation of secure user authentication.

2.5.2 Rich Authentication Features

The service not only handles user authentication but also manages user accounts and sessions with ease. It's equipped with features such as email and password reset, account verification, and sign-in link capabilities, which contribute to a robust authentication flow.

2.5.3 Security and Compliance

Firebase Authentication ensures that user credentials are securely managed and stored. The platform is compliant with identity standards such as OAuth 2.0 and OpenID Connect, so user data is handled in a secure and standardized manner.

2.5.4 Scalability and Flexibility

Designed to handle large-scale applications, Firebase Authentication can effortlessly scale to accommodate millions of users. The flexibility offered by Firebase allows developers to focus on the user experience while Firebase takes care of the authentication backend.

2.5.5 Integration with Firebase Ecosystem

Firebase Authentication is part of the larger Firebase ecosystem, which means it can be easily integrated with other Firebase services like Firestore, Firebase Realtime Database, and Firebase Cloud Functions. This integration provides a seamless development experience and allows for the creation of sophisticated, authenticated workflows.

By incorporating Firebase Authentication, the application streamlines the user sign-in process and ensures that authentication is handled safely and efficiently, allowing developers to focus on building features that add value to the user experience rather than the intricacies of user security management.

3 Design

The architecture of our voting application is built upon a microservice framework, utilizing Spring Boot for the backend services and React for the frontend user interface. This section outlines the high-level architecture and implementation details of the application.

3.1 Client-Side (Frontend)

The frontend is implemented using React, which provides:

- A dynamic and responsive user interface for end-users.
- Modular components that communicate with the backend via RESTful APIs.

3.2 Server-Side (Backend)

Powered by Spring Boot, the backend is composed of:

- Microservices that handle specific business capabilities, running independently and communicating through RabbitMQ for messaging.
- A layered architecture comprising Controllers, Services, Repositories, and Models.

3.2.1 Application Layers

The backend is divided into several layers, each responsible for distinct roles within the application:

- **Controller Layer**: Manages HTTP requests and responses.
- **Service Layer**: Orchestrates business logic and data flow.

- **Repository Layer**: Provides data access to the database.
- **Model Layer**: Represents the application's domain model.

3.2.2 Data Management and Storage

Data persistence is achieved through:

- JPA with Hibernate for object-relational mapping and database interactions.
- An H2 in-memory database for development, ensuring speed and ease of configuration.

3.2.3 Security and Authentication

Firebase Authentication is integrated to provide:

- Secure authentication mechanisms including social logins and OAuth2 flows.
- An abstraction layer for security concerns, allowing focus on core functionalities.

3.2.4 Real-Time Messaging and Updates

The integration of Dweet.io facilitates:

- Real-time messaging capabilities, particularly beneficial for IoT scenarios.
- Immediate data update dissemination to clients.

3.3 Conclusion

The application leverages the strengths of its chosen technologies to provide a robust, scalable, and performant system. The architecture ensures a clean separation of concerns, promoting maintainability and flexibility to cater to the evolving requirements of the application.

About 4 pages on:

1. An architectural overview of the application that has been implemented
2. High-level design, domain model, ... (App assignment A)
3. May involve selected models from Chaps. 5 of the IoT and cloud books

The example below shows how you may include code. There are similar styles for many other langages - in case you do not use Java in your project. You can wrap the listing into a figure in case you need to refer to it. How to create a figure was shown in Section 2.

```
1  public class BoksVolum {
2
3     public static void main(String[] args) {
4
5        int b, h, d;
6        String btext, htext, dtext;
7
8        [ ... ]
9
10       int volum = b * h * d;
11
12       String respons =
13             "Volum [" + htext + "," + btext + "," + dtext + "] = " + volum;
14
15    }
16 }
```

4 Prototype Implementation

This section should provide details of how the prototype has been implemented which may involve presentation of suitable code snippets.

5 Test-bed Environment and Experiments

About 2 pages that:

Explains how the prototype has been tested the test-bed environment.

Explains what experiments have been done and the results.

For some reports you may have to include a table with experimental results are other kinds of tables that for instance compares technologies. Table 1 gives an example of how to create a table.

6 Conclusions

Concludes on the project, including the technology, its maturity, learning curve, and quality of the documentation.

Config	Property	States	Edges	Peak	E-Time	C-Time	T-Time
22-2	A	7,944	22,419	6.6 %	7 ms	42.9%	485.7%
22-2	A	7,944	22,419	6.6 %	7 ms	42.9%	471.4%
30-2	B	14,672	41,611	4.9 %	14 ms	42.9%	464.3%
30-2	C	14,672	41,611	4.9 %	15 ms	40.0%	420.0%
10-3	D	24,052	98,671	19.8 %	35 ms	31.4%	285.7%
10-3	E	24,052	98,671	19.8 %	35 ms	34.3%	308.6%

Table 1: Selected experimental results on the communication protocol example.

The references used throughput the report should constitute a well chosen set of references, suitable for someone interesting in learning about the technology.

References