Technical Report - **Product specification**

BoardGamesHub

Course:	IES - Introdução à Engenharia de Software
Date:	Aveiro, 20/13/2023
Students:	105937: Bárbara Nóbrega Galiza 107969: João Miguel Dias Andrade 109018: Tomás António de Oliveira Victal 109986: Pedro Daniel Fidalgo de Pinho
Project abstract:	BoardGamesHub is a web application for comparing board games prices across multiple web stores, providing their current price and showcasing their price history. It also serves as a discover for new board games.

Table of contents:

1 Introduction

This project was made in the scope of the class “Introduction to Software Engineering”, with the intention of learning and applying software engineering best practices. In this case, the proposed method was the agile process, alongside with the use of user stories and scenarios, and the theme was given by the students. Our product choice was a web application for board games price comparison, the BoardGamesHub.

This report describes the product concept, the architecture and the information model in which the system is based on.

2 Product concept

Vision statement

The BoardGamesHub application is intended to be used by board games players to find the lowest price of a certain game across the offers of many vendors. This idea was based on already existent websites for price comparison, like “KuantoKusta” and “Trivago”, with the difference being the scope of the product.

The choice of the scope “Board games” was made based on the fact of it being a not very well explored subarea within the price comparison area, and it being a business domain known by one of the team members.

In that sense, alongside with the price comparison, our website showcases detailed information about the games, a price history to assist the user on when to buy it, specific filters functionalities to help the user find new games, a wishlist, notifications for discounts and recommended games based on user interests.

Personas and Scenarios

John is our common user, he is 47 years old, male, and lives in California with his wife.

He's a big board game collector and always wants to acquire new interisting games.

Due to this hobby being very expensive, he requires a way to help him keep growing his collection with a somewhat tight budget.

Matilda is our administrator, she is 36 years old, female, and is currently engaged.

She's a very busy person due to running multiple businesses, so her time is short. She wants things done fast and efficiently.

Her occupation as a manager doesn't require any porgramming skill, so she needs a straightfoward. UI-based way to manage the application information.

Scenario 1:

John has been looking forward to buying "Voidfall" but has been waiting for a sale to come around. He then access the BoardGamesHub application and receives a notification, which says "Voidfall", which he had previously wishlisted, is now on sale on Amazon.

Scenario 2:

Matilda heard a complaint from a company that they do not wish to have their game displayed on our frontend. She is quick to act, so she signs in with her admin credentials, access the “delete” tab and deletes the given game.

Product requirements (User stories)

John (User):

Epic: Finding the best price for a specific game.

- As John, I want to see all the prices for a game across multiple stores, so that I can buy the game for the lowest price there is.
- As John, I want to consult the price history of a board game, so that I can identify the best time to buy it.

Epic: Searching and comparing various games.

- As John, I want to have access to a list of all board games, along with a search engine and filters to help me find games within certain categories.
- As John, I want to have recommendations based on the games I play/look at so that I can find similar games with a better price.
- As John, I want to edit my account preferences (board game categories) so that my recommended games will only include games within these categories.
- As John, I want to search for board game publishers to help me find all the games from some specific publisher.

Epic: Wishlist managment.

- As John, I want to be able to create a wishlist, so that I can keep track of all the board games I wish to get.
- As John, I want to be able to update my wishlist, in order to add new games that interest me and remove games I’m no longer interested in.
- As John, I want to receive a notification informing me that one of my games in my wishlist is currently on sale, so that I can buy it for a lower price.

Matilda (Admin):

Epic: Website managment.

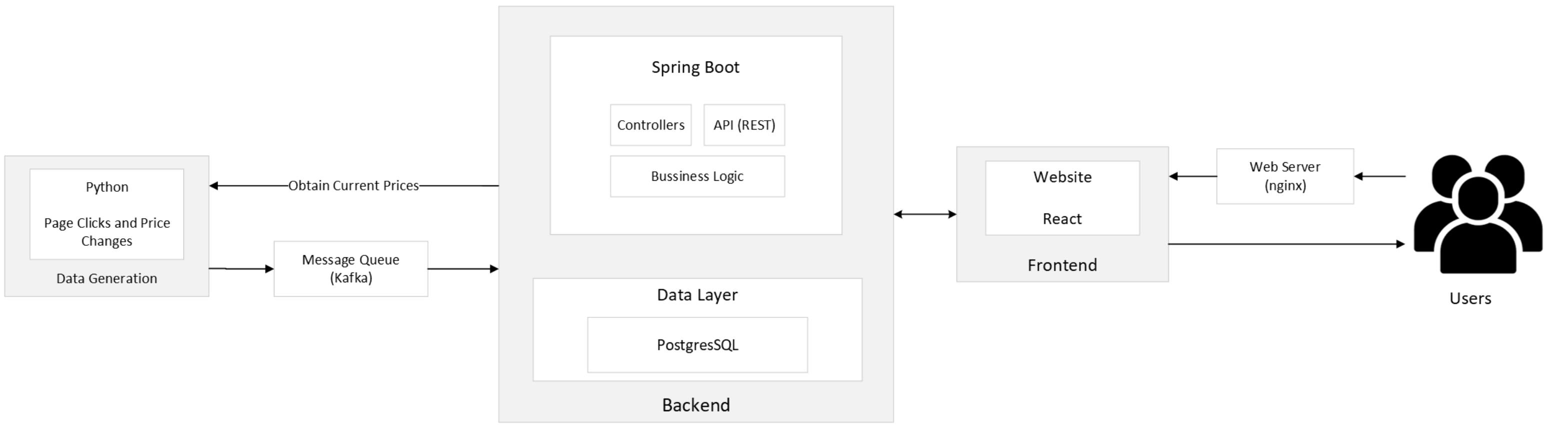
- As Matilda, I want to be capable of adding a new game to be tracked in that site, in order to give more game options to the clients.
- As Matilda, I want to be capable of removing a game uppon vendor request, so that the site follows the business rules.

3 Architecture notebook

Key requirements and constrains

- The data sent from the data sources must not be lost in any situation. In case of a failure this data must be saved.
- All the prices changes must be stored so that the users are able to view the price history for a product.
- An admin must be able to, when adding a new board game, fill the information about the game.
- All the data related to the website (users, games, publishers, artists, clicks, etc) must be stored in a consistent manner.

Architetural view



In the Frontend, we used ReactJS since it allows use to separate the frontend from the backend. React can also easily consume the RESTful APIs created in Spring Boot. To allow the users to reach the website, we used nginx as our web server.

For the Backend, we used Spring Boot since it allows us to easily implement the controllers, the API, the business logic and to easily get the website running. In the Data Layer we have a database where we store all the information about the games, the users and the prices. For our database we settled on PostgreSQL since it is one of the most popular relational databases.

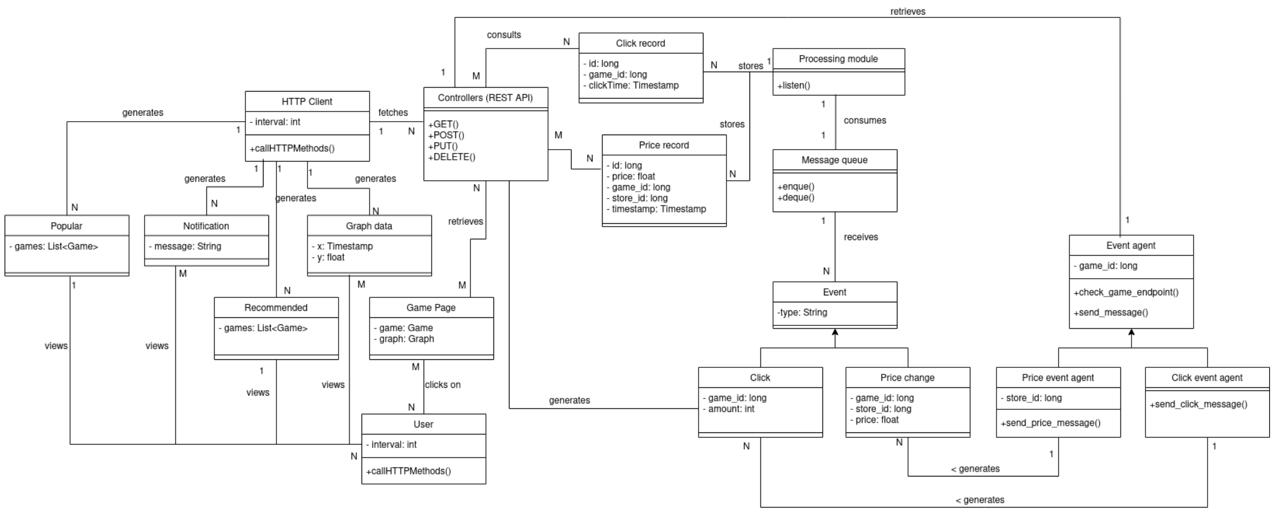
Our data sources emulate the fluctuations in prices and user “clicks” or visits to a certain products’ page. This allows the user to see price changes and, through the use of the wishlist, be notified when a game goes on sale. To pass this information, we used Kafka. We chose Kafka because it has data persistence, making it so that information doesn’t get lost if the service goes down.

Module interactions

The frontend communicates with the controllers in the backend. Information like price changes are sent to the frontend and, for example, are shown as notifications in case the user has the game in the wishlist.

Our data sources retrive the current princes from the backend through the API and generate new prices to simulate price changes. It also simulates user actions. The new data is then sent back to the backend using Kafka as the message queue.

4 Information perspetive



- Event Agent: The event agent is the data source for our system. It generates both prices and clicks. For the prices, it retrieves the current price for a game and store from the API, generates discounts or increases, and sens the new price to the message queue.
- Processing Module: The processing module is simply the consumer for the message queue, that was created as a java class. It receives the events and stores then on each respective table on the database.
- Controllers (REST API): The API controllers consult the tables when called from the HTTP Client, on the frontend, and return the updated values. The controllers also play a role at user generated data, when user access the game page (generates a Click).
- HTTP Client: The HTTP Client and frontend logic generate the UI to indicate that data has changed. This can be visualized via four components: notifications, recommended and popular lists and the price history graph for each game.
- User: The user can visualize the generated data and also generate data by clicking on games.

5 References and resources

Key Components:

Frontend:

https://axios-http.com/

https://react.dev/

https://reactrouter.com/en/main

https://tailwindcss.com/

https://www.framer.com/motion/

https://www.docker.com/

https://docs.pmnd.rs/zustand/getting-started/introduction

https://nivo.rocks/

https://react-icons.github.io/react-icons/

https://www.figma.com/

Backend:

https://kafka.apache.org/

https://spring.io/projects/spring-boot/

https://spring.io/projects/spring-security/

https://projectlombok.org/

https://jwt.io/

https://springdoc.org/

https://www.postgresql.org/

https://spring.io/projects/spring-data-jpa/

Key references:

https://www.baeldung.com/