IT 309 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

Auctions-ba

Prepared by:
Amina Kodžaga
Ajdin Hukić

Proposed to:
Nermina Durmić, Assist. Prof. Dr.
Aldin Kovačević, Teaching Assistant

16.06.2023.

Table of contents

1. Introduction

Through this document we will present our work on project named Auctions-ba. We will describe in more details what we did so far.

1.1. About the Project

We created an application named Auctions-ba that facilitates the buying and selling of goods or services through a competitive bidding process. The project focuses on creating a system for sellers to list their items or services for auction, including setting up descriptions, images, starting prices and other relevant details.
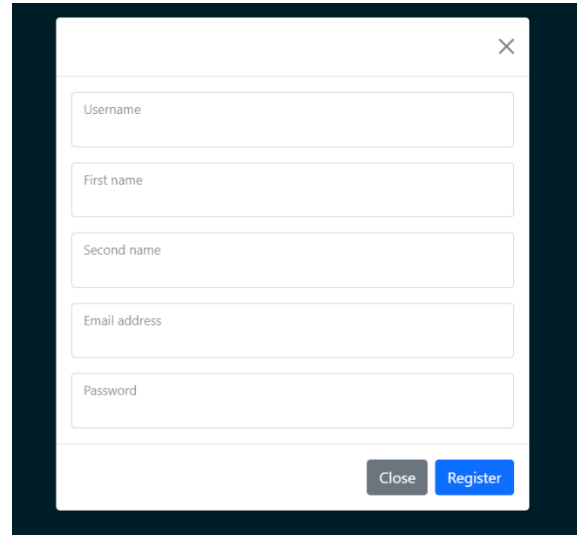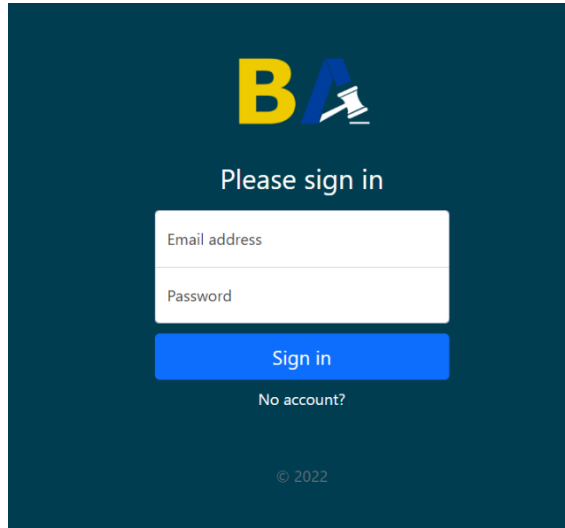
Link where our project is deployed: https://keytrackedu.com/auctions-ba/

1.2. Project Functionalities and Screenshots

Describe or list the main features of the application and provide a few screenshots of your project.

Main features:

- Login and registration

- Bids

- My item

- Add item

Welcome! It's time to bid.

6 DAYS LEFT

Mouse

Highest bid 5BAM

8 DAYS LEFT

Airforce

Highest bid 33BAM

13 DAYS LEFT

Macbook

Highest bid 3BAM

---

6 DAYS LEFT

6 DAYS LEFT

G203

↩

Mouse

ⓘ

Prodajem nov miš sa garancijom.
Kupljen u americi dok se bilo na
vekejšnu i sad treba da se zaradi
para nešto na ovoj sirotinji iz bosne.
Nije nikakav skem u pitanju čisto da
odma naglasim nisam ja davud.

Bid 5BAM or more

Mouse

Enter bid amount   **Submit**

Highest bid 5BAM   **Highest bid 33BAM**   **Highest bid 3BAM**

Add item to auction ✕

Title:

Description:

Image:

Choose File   No file chosen

Ending date:

06/16/2023 08:56 PM

Add item

2. Project Structure

2.1. Technologies

In this project we used technologies as PHP and framework Flight PHP for backend and HTML, JS and CSS for frontend.

Coding standards that we used are JavaScript standard code style and PSR-12.

Architecture that we used:

Model-View-Controller (MVC): The project follows the MVC pattern, where the routes defined in index.php act as the controllers, handling the requests and invoking appropriate services and DAOs. The services and DAOs represent the model layer, responsible for business logic and data access, respectively. The responses from the controllers are sent to the client as views.

Dependency Injection (DI): The project utilizes DI to inject dependencies into various components. For example, the Flight framework is used for routing, and dependencies like the user DAO and JWT configuration are injected into the controllers and services. This promotes loose coupling and improves testability and maintainability.

Data Access Object (DAO): The DAO pattern is employed to separate the data access logic from the rest of the application. The UserDao class provides methods to interact with the underlying database for user-related operations.

Repository: Although not explicitly defined in the provided code, the DAO classes can be considered as a form of repository pattern. They encapsulate the data access logic, providing a clean interface to interact with the data source.

RESTful API: The project follows the principles of REST (Representational State Transfer) to design the API endpoints. It uses HTTP methods (POST, GET) to perform CRUD (Create, Read, Update, Delete) operations on user data. The endpoints are structured in a resource-oriented manner, providing a stateless interface to access and manipulate user-related information.

2.2. Database Entities

Tables in database:

- Bids - proposal made by individuals

- Items – elements that should be sold

- Users – someone who uses the application and want to buy or sell something

2.3. Design Patterns

List the *design patterns* that we used in the project:

1. Singleton Pattern: The Config class in index.php is implemented as a Singleton pattern. It ensures that only one instance of the Config class is created and provides a global point of access to the configuration settings throughout the project.

2. Factory Method Pattern: The Flight class in index.php acts as a Factory Method pattern. It creates instances of various classes such as UserService, ItemService, BidService, and UserDao, which are used throughout the application.

3. Dependency Injection: The Flight class in index.php demonstrates the use of Dependency Injection by creating instances of services and injecting them into routes. For example, UserService, ItemService, and BidService instances are injected into various routes to handle user-related, item-related, and bid-related functionality, respectively.

4. Data Access Object (DAO) Pattern: The UserDao class in dao.php represents the Data Access Object pattern. It provides methods for interacting with the underlying database and encapsulates the database operations related to the users table.

5. Service Layer Pattern: The UserService, ItemService, and BidService classes in services.php act as service classes, implementing the Service Layer pattern. They provide higher-level, domain-specific functionality and encapsulate the business logic for user-related, item-related, and bid-related operations, respectively.

6. Builder Pattern: The Flight::json() method used in various routes, such as POST /register, POST /login, POST /item, DELETE /item/{id}, etc., can be considered as an implementation of the Builder pattern. It constructs and returns JSON responses using a fluent interface, allowing flexibility in building the response object.

7. Front Controller Pattern: The index.php file serves as the entry point and acts as a Front Controller for the application. It receives all requests, initializes the necessary components, handles routing, and dispatches requests to appropriate routes.

2.4. Tests

PHPUnit tests

testRegisterUserEmail: This test verifies the registration process by attempting to register a user with an already registered email address. It expects a response with a status code of 500 and the message "Email already registered."

testRegisterUserUserName: This test checks the registration process by attempting to register a user with an already taken username. It expects a response with a status code of 500 and the message "Username already registered."

testLoginUser: This test validates the login functionality by attempting to log in with an email address that does not exist in the system. It expects a response with a status code of 404 and the message "User doesn't exist."

testLoginWrongPasswordUser: This test ensures that the login process rejects incorrect passwords by attempting to log in with a valid email address but an incorrect password. It expects a response with a status code of 404 and the message "Wrong password."

testLoginTrueUser: This test verifies the successful login process by attempting to log in with a valid email address and password. It expects a response with a status code of 200 and a token in the response data.

3. Conclusion

Through the implementation of this project we learned more about architectural and design pattern and coding styles. Process of learning was interesting, not so difficult. We think that we implemented useful application which satisfies all requirements.