Geekbrains

**Разработка веб-приложения «Мир Настольных Игр»   
на Java Spring**

Программа:

Веб-разработка на Java

Дорогова Е.Б.

Москва

2025

**Содержание**

[Введение 3](#_Toc188561416)

[Глава I. Предпроектная стадия разработки 6](#_Toc188561417)

[1.1. Алгоритм работ 6](#_Toc188561418)

[1.2. Формирование функциональных требований 7](#_Toc188561419)

[1.3. Определение набора используемых технологий 8](#_Toc188561420)

[1.4. Архитектура приложения 9](#_Toc188561421)

[Глава II. Основная стадия разработки проекта 12](#_Toc188561422)

[2.1. Разработка моделей данных 12](#_Toc188561423)

[2.2. Разработка сервисного слоя 14](#_Toc188561424)

[2.3. Разработка контроллера для управления приложением 20](#_Toc188561425)

[Глава III. Базы данных 30](#_Toc188561426)

[Глава IV. Обеспечение безопасности проекта 40](#_Toc188561427)

[Глава V. Docker 47](#_Toc188561428)

[Глава VI. Применение шаблонизатора Thymeleaf 49](#_Toc188561429)

[Заключение 51](#_Toc188561430)

[Список используемой литературы 54](#_Toc188561431)

# Введение

Ввиду своего увлечения настольными играми я всё больше знакомилась с людьми, которые разделяют моё хобби, и пришла к выводу, что круг любителей настольных игр неустанно растёт. Помимо коллекционирования красивых коробок и получения удовольствия от сыгранных партий существует потребность ведения краткой статистики и учёта своих настольных игр.   
По этой причине мной было принято решение разработать веб-приложение, которое поможет удовлетворить вышеуказанные потребности.

Веб-приложение «Мир настольных игр» позволяет увлекающимся настольными играми пользователям вносить данные о своих приобретениях,   
а также представлять свою коллекцию на всеобщее обозрение, вносить данные о партиях, сыгранных в определённую настольную игру с указанием краткой характеристики.

С помощью данного приложения любители настольных игр могут легко вернуться и вспомнить сыгранные партии, сформировать мнение о настольной игре, а также поддерживать порядок в своей коллекции. Веб-приложение даёт возможность находить для себя новинки благодаря коллекциям других пользователей, а с помощью информации о партиях подбирать оптимальное количество игроков, ведь это напрямую влияет на то, как раскроется каждая партия.

Несмотря на простое оформление веб-приложения, его интерфейс интуитивно понятен, и пользователи без труда смогут вносить необходимые записи и просматривать информацию других пользователей.

Из вышесказанного следует, что веб-приложение «Мир Настольных Игр» обладает следующим функционалом:

* **Записи о настольных играх** – позволяет пользователю вносить информацию о настольных играх с указанием их жанра и даты появления   
  в коллекции.
* **Записи о партиях** – в каждой записи о настольной игре пользователь имеет возможность вести учёт сыгранных партий, в котором содержится вся необходимая информация, которая позволит провести краткий анализ.
* **Редактирование и удаление записей** – необходимый инструмент, который позволит актуализировать информацию и внести уточнения. Данный инструмент применим к записям и о настольных играх, и партиях.
* **Просмотр коллекций и партий других пользователей** – то,   
  без чего невозможно представить жизнь «настольщика», ведь это является важным элементом хобби, благодаря которому пополняется коллекция   
  и, например, вносятся корректировки в собственный игровой процесс.

При разработке веб-приложения использовались такие инструменты как:

* **Postman** – для предварительного тестирования правильной настройки связей между классами и работы методов, перед тем как приступить к настройке контроллера для работы с html-страницами.
* **База данных H2** – применялась на ранних стадиях разработки приложения для упрощения работы с таблицами данных. Проста для установки и в использовании.
* **Git –** данная система создана для контроля версий, она позволяет чётко отслеживать все вносимые в проект изменения, а также   
  при необходимости возвращаться в любое из ранее сохранённых состояний   
  и продолжить разработку с указанной точки.
* **Docker –** для удобства работы с базами данных, а также «портативности» приложения и упрощения его запуска в других средах.

С целью защиты конфиденциальности и безопасности пользователей в веб-приложении «Мир Настольных Игр» используется простая система безопасности, которая с помощью пароля позволяет защитить данные, а также ограничить к ним доступ.

**Задачи проекта:**

1. Проанализировать потребности пользователей в управлении коллекцией настольных игр.
2. Спроектировать архитектуру веб-приложения.
3. Реализовать приложение с использованием Spring Boot, Hibernate,   
   и других инструментов.
4. Обеспечить удобный интерфейс для взаимодействия с системой.
5. Провести тестирование и анализ производительности приложения.

Веб-приложение «Мир Настольных Игр» было разработано студентом GeekBrains по направлению обучения «Разработчик. Техническая специализация: Веб-разработка на Java» Дороговой Евгенией Борисовной.

# Глава I. Предпроектная стадия разработки

# Алгоритм работ

На предпроектной стадии был выполнен анализ предметной области   
и выявлены основные требования, осуществлены предварительные работы. Данный этап включал в себя следующие действия и работы:

1.1.1. Изучение предметной области.

Был проведён обзор существующих решений для управления коллекциями настольных игр. В ходе исследования выяснилось,   
что большинство популярных приложений либо не поддерживают управление игровыми партиями, либо информация о партиях является недостаточной, либо эти приложения обладают избыточной функциональностью, которая усложняет работу пользователя и отталкивает от дальнейшего использования. Это подтвердило необходимость создания нового приложения, которое ориентировано на простое, удобное, а также оптимальное использования приложения.

1.1.2. Сбор и анализ требований.

Для определения требований к системе был проведён опрос среди любителей настольных игр, чтобы выявить их потребности и предпочтения. Основные запросы включали возможность добавления игр, ведения учёта игровых партий и их редактирование.

1.1.3. Техническое обоснование.

В качестве платформы разработки выбрана Java с использованием Spring Framework. Данное решение обеспечивает надёжность, масштабируемость   
и удобство разработки.

1.1.4. Генерация проекта.

С помощью SpringInitializr был создан новый проект, который послужит основой для веб-приложения. Для проекта выбраны и добавлены требующиеся зависимости (SpringWeb, Lombok, SpringSecurity, SpringDataJPA, Thymleaf   
и H2 Database, которая впоследствии была заменена на MySQL Connector   
для работы с базой данной MySQL, подключённой через Docker).

1.1.5. Разработка структуры приложения.

Была спроектирована базовая архитектура приложения, включающая следующие компоненты:

**База данных**: Таблицы для хранения пользователей, настольных игр   
и игровых партий;

**Серверная часть**: REST API на основе Spring Boot.

# Формирование функциональных требований

На основании проведённого анализа были сформированы следующие функциональные требования к системе:

* Должна присутствовать возможность внесения записей   
  о настольных играх и сыгранных партиях.
* Функционал, позволяющий редактировать и удалять записи   
  об играх и партиях.
* Необходима общая сводка информации по партиям, для получения актуальной информации о статистике побед и популярности игр, а также возможность просматривать партии определённого пользователя   
  по конкретной настольной игре.
* Ведение истории игровых партий с указанием победителей, количества игроков и комментариев.
* Регистрация и авторизация пользователей.
* Простота в пользовании приложением, т.е. любой должен интуитивно понимать функционал веб-приложения.
* Обеспечение безопасности данных пользователей, которое подразумевает под собой защиту от несанкционированного доступа к данным.
* Аутентификация и авторизация – доступ к данным и отсутствие возможности манипулировать данными других пользователей.
* Веб-приложение должно быть масштабируемым, то есть присутствует возможность расширения функционала, совершенствования имеющихся методов, а также роста числа пользователей.
* Наличие поддержки приложения – работы, направленные   
  на совершенствование работы веб-приложения и безопасности.

# Определение набора используемых технологий

Для разработки веб-приложения «Мир Настольных Игр» был выбран перечень следующих технологий:

Язык программирования:

Для разработки серверной части выбрана Java, которая обеспечивает стабильность, масштабируемость и надёжность веб-приложений.

Фреймворк для веб-разработки:

Используется Spring Boot, включающий модули Spring MVC и Spring Security. Spring MVC предоставляет удобные инструменты для построения интерфейсов и обработки запросов, а Spring Security гарантирует высокий уровень безопасности.

База данных:

MySQL была выбрана для хранения данных о пользователях, настольных играх и партиях. Эта база данных отличается высокой производительностью, надёжностью и возможностью масштабирования. MySQL также характеризуется высоким уровнем гибкости, то есть она легко подстраивается под потребности разработчиков. Немаловажным плюсом является то, что доступ к этой базе данных бесплатный.

Фронтенд:  
Для создания интерфейса приложения применяется HTML.

Шаблонизатор:  
Шаблонизатор Thymeleaf применяется для генерации динамических   
HTML-страниц на стороне сервера. Этот инструмент легко кооперируется   
со Spring Boot и позволяет удобно отображать данные на языке Java.

Аутентификация и авторизация пользователей:

Для управления доступом и обеспечения безопасности в приложении применяется Spring Security.

Среда разработки:

IntelliJ IDEA выбрана как наиболее удобная среда разработки для создания веб-приложений. Также во время обучения на GeekBrains была практика работы в Visual Studio Code, который значительно проигрывает   
в удобстве и функциональности использования. Помимо удобства IntelliJ IDEA предоставляет широкий спектр инструментов для работы   
с Java-приложениями.

# Архитектура приложения

Веб-приложение «Мир Настольных Игр» построено на основе архитектурного паттерна MVC (Model-View-Controller), который обеспечивает разделение ответственности на 3 модуля: данные, представление и логика. Таким образом блоки кода будут независимыми, их можно менять без ущерба другим блокам. Например, при изменении способа обработки данных, и при этом способ их отображения не будет затронут. MVC характеризуется простотой для разработки и поддержки приложений.

На основании архитектурного паттерна MVC структура веб-приложения «Мир настольных игр» включает:

Модель / Model:

Этот компонент отвечает за представление данных приложения. Он включает классы, описывающие ключевые сущности: Пользователь, Настольная игра и Партия. Также Модель реагирует в соответствии   
с командами, направляемыми Контроллерами. То есть Модели передают запрашиваемую информацию или изменяют своё состояние.

Представление / View:

Отвечает за отображение данных, как их видит пользователь. Thymeleaf используется для создания динамических HTML-шаблонов, которые получают данные от контроллеров.

Контроллер / Controller:

Контроллер управляет взаимодействием между пользователем, моделью и представлением. Методы контроллеров, аннотированные @Controller   
или @RestController, обрабатывают запросы (GET, POST, PUT, DELETE), вызывая соответствующие сервисы для выполнения бизнес-логики. Контроллер определяет, как веб-приложение будет реагировать на действия пользователя. Также контроллер может отвечать за фильтрацию данных   
и авторизацию.

Репозиторий / Repository:

Репозиторий обеспечивает доступ к данным в базе данных. Используются интерфейсы, расширяющие JpaRepository, для работы   
с запросами на получение, сохранение, обновление и удаление данных.

Сервис / Service:

Сервисы реализуют бизнес-логику приложения, координируют действия между компонентами и обеспечивают единый доступ к функциональности. Однако к сервису применяется ряд условий для его работы:

* Сервис не может самостоятельно обращаться к контроллерам;
* Сервис для своего функционирования обращается к базам данных, моделям и сущностям.

Сервер / Server:

Приложение запускается на встроенном сервере, таком как Tomcat   
или Jetty, который обрабатывает HTTP-запросы и управляет жизненным циклом приложения.

Эти компоненты обеспечивают модульность, гибкость   
и масштабируемость системы, позволяя разрабатывать и тестировать части приложения независимо друг от друга, что обеспечивает более эффективный подход к разработке приложения.

# Глава II. Основная стадия разработки проекта

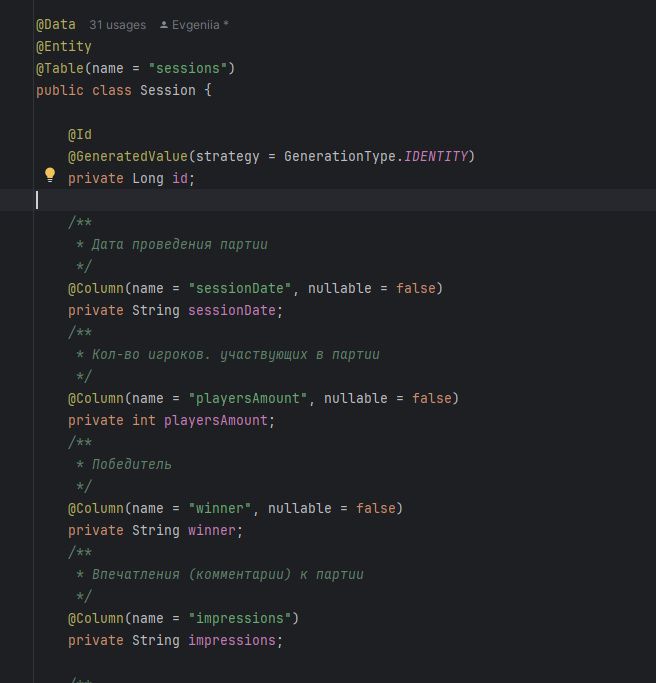
# 2.1. Разработка моделей данных

Разработка веб-приложения началась с разработки первого модуля шаблона проектирования MVC – Модели.

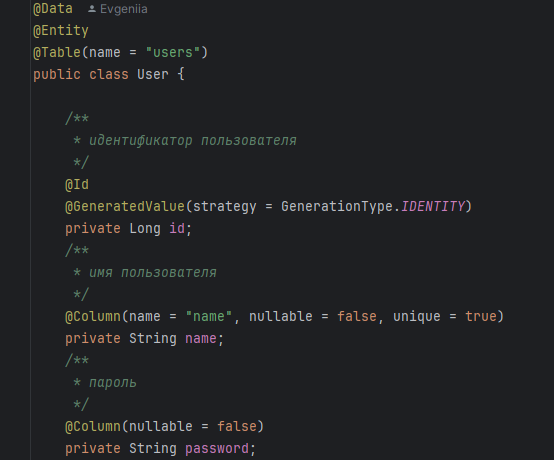
Основной класс модели BoardGame будет связан с другими двумя сущностями и представляет собой запись о настольной игре пользователя, которая содержит название настольной игры (title), жанр, в котором она представлена (genre) и дату добавления в коллекцию (gettingDate).



Следующий класс Session включает в себя информацию о партиях настольной игры с указанием даты проведения партии (sessionDate), количества игроков (playersAmount), победителя партии (winner) и краткая сводка   
по впечатлениям (impressions). Благодаря этим данным пользователи могут собирать информацию по настольным играм.



Сам пользователь представлен классом User. Как и предыдущих классах модели у пользователя есть генерируемый идентификатор, имя пользователя (name) и пароль (password). Пароль будет использоваться для аутентификации и авторизации.



Каждый класс помечен аннотациями JPA, такими как @Entity, @Id, @GeneratedValue, и включает дополнительные связи с использованием аннотаций @OneToMany и @ManyToOne.

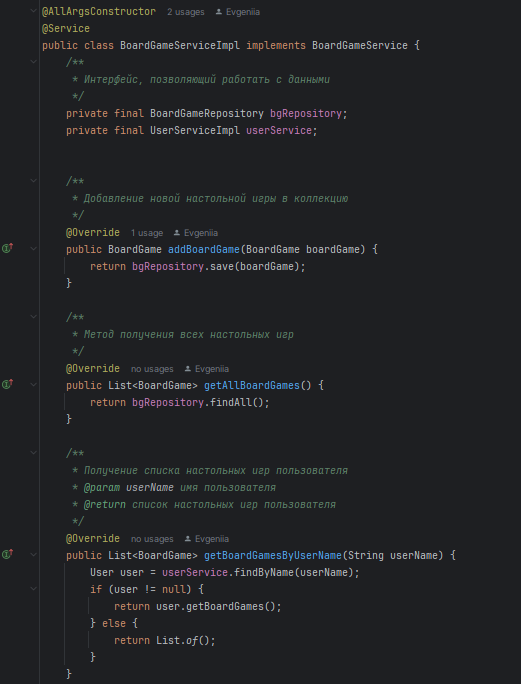
# 2.2. Разработка сервисного слоя

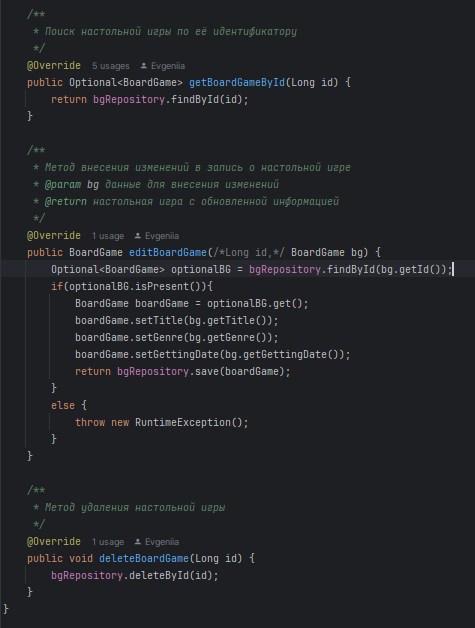
Логика приложения сосредоточена в сервисах: UserServiceImpl, BoardGameServiceImpl, SessionServiceImpl. Однако для установки гибких связей созданы 3 интерфейса, которые позволяют сервисам и контроллерам взаимодействовать вне зависимости от их реализации и облегчить интеграцию между друг другом.

Все 3 класса имеют аннотации:

* @Service – эта аннотация входит в SpringFramework и применяется   
  для отметки классов, которые выполняют логику, связанную с обработкой данных.
* @AllArgsConstructor – аннотация библиотеки Lombok, которая генерирует конструктор. В свою очередь этот конструктор принимает все поля класса в качестве аргументов. Он зачастую используется для внедрения зависимостей через конструктор.

BoardGameServiceImpl реализует сервис BoardGameService и вводит функционал, благодаря которому происходит управление настольными играми.



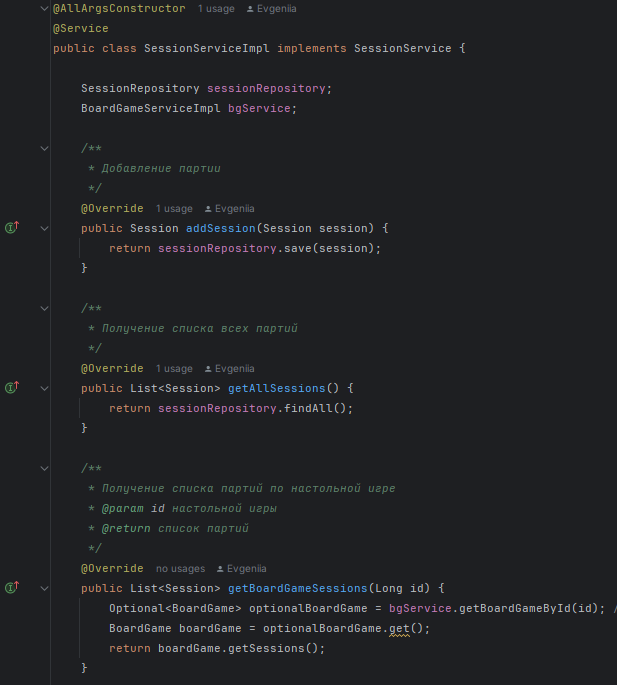


Методы addBoardGame и deleteBoardGame добавляют и соответственно удаляют записи о настольных играх.

Метод getAllBoardGames использовался для тестирования через Postman. Возвращает список всех игр из репозитория. А метод getBoardGamesByUserName уже активно используется в финальной версии веб-приложения и выводит список всех игр пользователя. То есть   
при осуществлении перехода по имени пользователя можно увидеть его коллекцию.

В методе editBoardGame осуществляется реализация возможности редактирования записей о настольных играх (изменить название, жанр или дату появления в коллекции).

В классе SessionServiceImpl обрабатываются данные по прошедшим партиям.



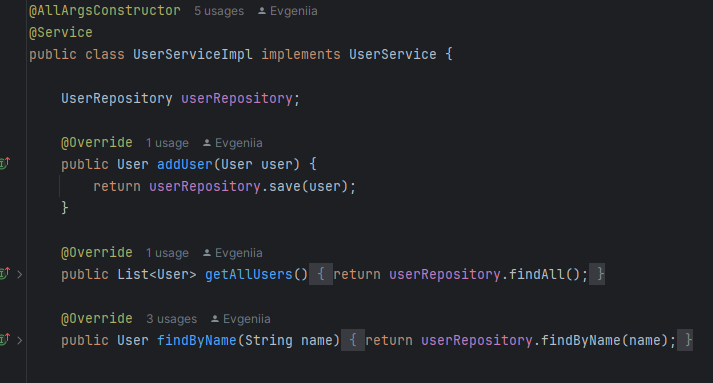


В этом сервисе также представлены методы по управлению данными: добавление, редактирование, удаление.

Метод getAllSessions используется для сводки по партиям на главной странице веб-приложения, чтобы пользователи могли оперативно и удобно получать информацию об играх, в которые играют в настоящее время.

С помощью метода getBoardGameSessions осуществляется вывод всех партий по конкретной настольной игре определённого пользователя. Таким образом можно собрать статистику игрока.

Для работы с пользователем используется класс UserServiceImpl.

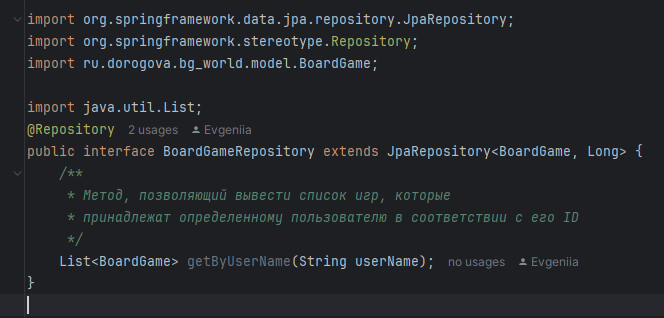


В этом классе реализуется возможность добавления нового пользователя, поиск пользователя по имени для аутентификации и авторизации.

Каждый сервис взаимодействует с соответствующими репозиториями, реализованными с помощью JpaRepository, для выполнения операций с базой данных.

Репозитории обеспечивают взаимодействие с базой данных, предоставляя удобный способ выполнения CRUD-операций. В проекте используются интерфейсы, расширяющие JpaRepository, что позволяет автоматически генерировать основные методы для работы с данными.

Репозитории упрощают работу с базой данных за счёт предопределённых методов, таких как save, delete, findById, и позволяют добавлять собственные запросы при необходимости. Например, метод getByUserName в BoardGameRepository был реализован для выборки данных о настольных играх конкретного пользователя.



# 2.3. Разработка контроллера для управления приложением

Контроллеры играют ключевую роль в управлении взаимодействием между пользователями, данными и бизнес-логикой. Они принимают входящие HTTP-запросы, связанные с управлением записями о настольных играх   
и партиях, обрабатывают данные запросы и возвращают ответы.

Контроллеры в проекте используют аннотацию @Controller   
или @RestController для обработки веб-запросов. Все методы контроллеров непосредственно связаны с сервисным слоем, что обеспечивает изоляцию бизнес-логики от обработки HTTP-запросов.

Посредством методов контроллера пользователи получают возможность взаимодействовать с записями в коллекциях настольных игр и партиями   
по конкретным играм. Помимо этого, методы контроллера также работают   
с перенаправлением пользователей на страницы, на которых отображаются результаты по запросам.

Для целей обеспечения безопасности пользователей при использовании веб-приложения «Мир Настольных Игр» устанавливается взаимодействие контроллера и SpringSecurity (модуль безопасности приложения). Посредством данного взаимодействия при доступе к функционалу обеспечивается аутентификация и авторизация.

Методы контроллера обрабатывают различные HTTP-запросы. Так, методы с @GetMapping (GET) применяются для получения данных. Это самый популярный HTTP-метод, вызываемый при отправке запроса на сервер   
для перехода на очередную страницу.

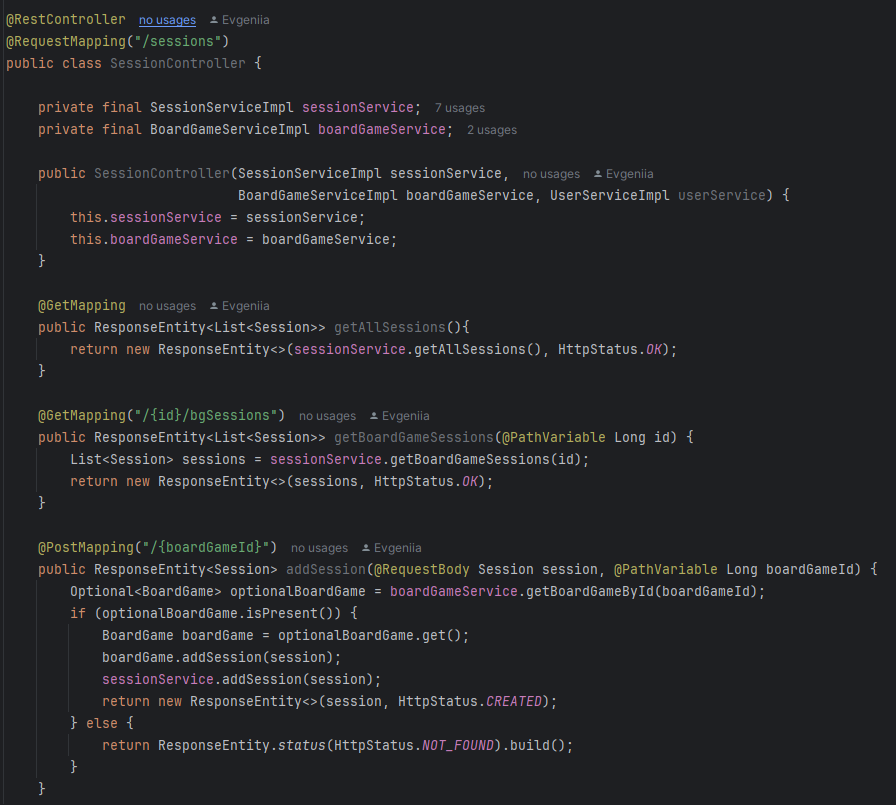
Метод POST заключается в передаче ресурса сервису, который должен создать объект.

PUT-запросы используются для изменения существующих объектов.

Для удаления ресурсов используются запросы DELETE.

В ходе разработки веб-приложения «Мир Настольных Игр»   
для тестирования изначально применялись контроллеры, созданные отдельно для каждой сущности для более удобного управления и тестирования. Эти контроллеры использовались для работы в Postman и были помечены аннотацией @RestController.

Вышеуказанная аннотация объединяет в себе аннотации @Controller   
и @ResponseBody и автоматически преобразует возвращаемые контроллером данные в формат JSON или XML. То есть подобные контроллеры возвращают исключительно данные, с их помощью нельзя получить страницы HTML   
или применить Thymleaf.



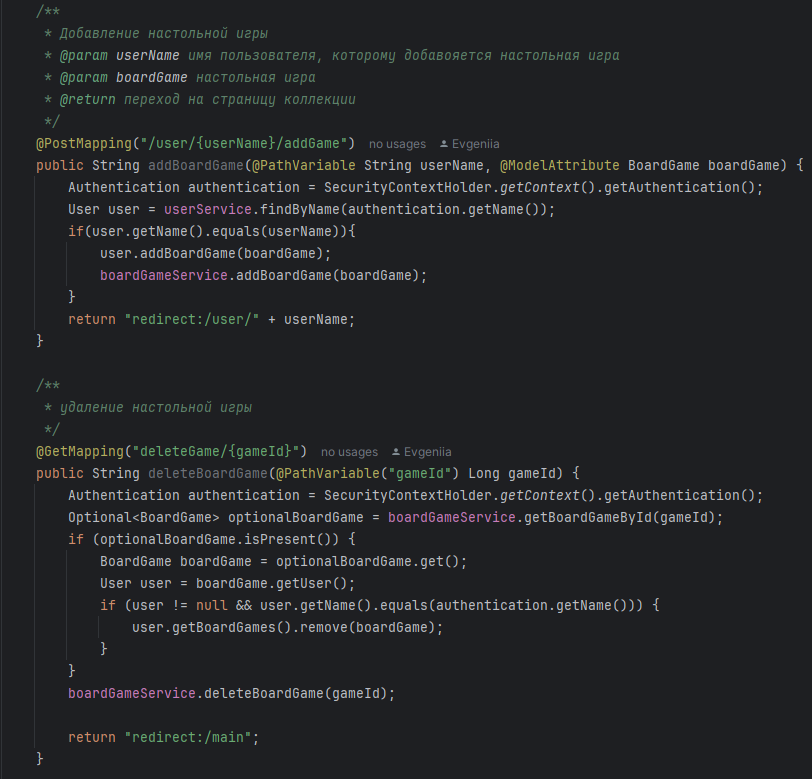
В дальнейшем для работы с HTML страницами был разработан единый контроллер, который работает и с настольными играми, и с партиями.

BGWorldController является неотъемлемой частью веб-приложения, которая служит для обеспечения удобного интерфейса, предназначенного   
для работы с запросами пользователей.



Первым методом, представленным в контроллере, является mainPage.   
С помощью методов сервисов, которые работают с пользователями и партиями, на странице отображается список пользователей, а также сыгранных партий   
с указанием названия настольной игры.

Посредством метода userCollection пользователи получают возможность перейти на страницы всех любителей настольных игр, зарегистрированных   
в веб-приложении, и ознакомиться с их коллекцией настольных игр. Данный метод применяет интерфейс сервера пользователя.



Далее начинаются методы, которые непосредственно взаимодействуют   
с объектами настольных игр. Так как экземпляр настольной игры непосредственно связан с пользователем и отображается на его странице,   
то сначала идёт определяется сам пользователь по имени, а затем ему присваивается настольная игра и к ней же применяется метод addBoardGame интерфейса сервиса настольных игр. Также стоит отметить, что с помощью аутентификации, предоставленной SpringSecurity, происходит проверка прав пользователя, что предотвращает возможность добавления игры другому пользователю.

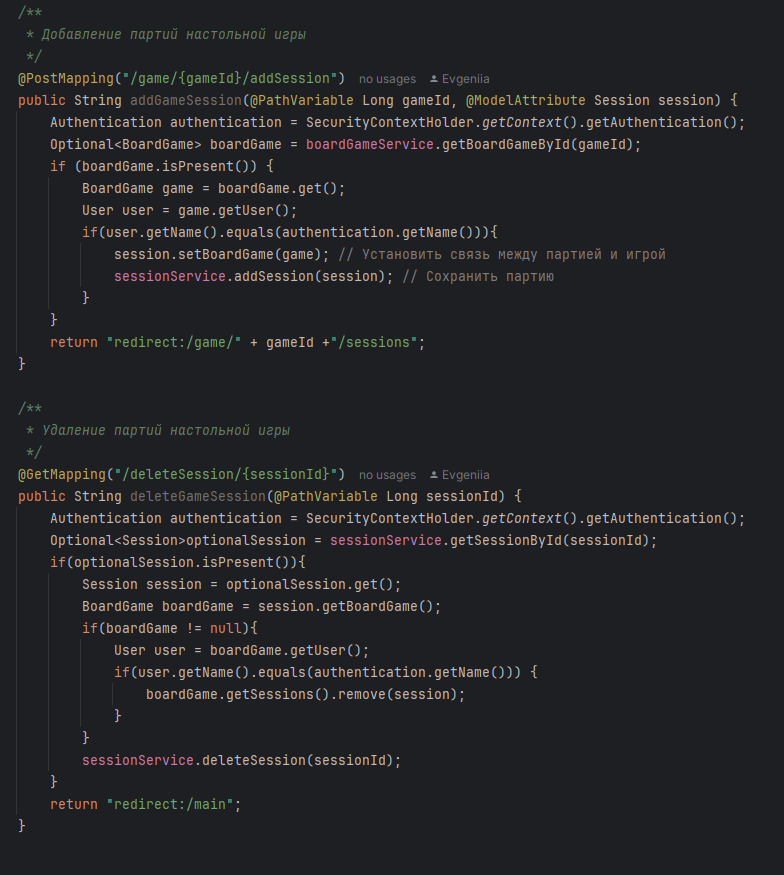
Для удаления записи о настольной игре необходим её идентификационный номер (ID), по которому проверяется действительно ли существует указанный экземпляр. Затем, как и в методе добавления настольной игры, проверяется пользователь, которому принадлежит данная запись, и тот, кто отправляет запрос на удаление. При совпадении происходит удаление записи.

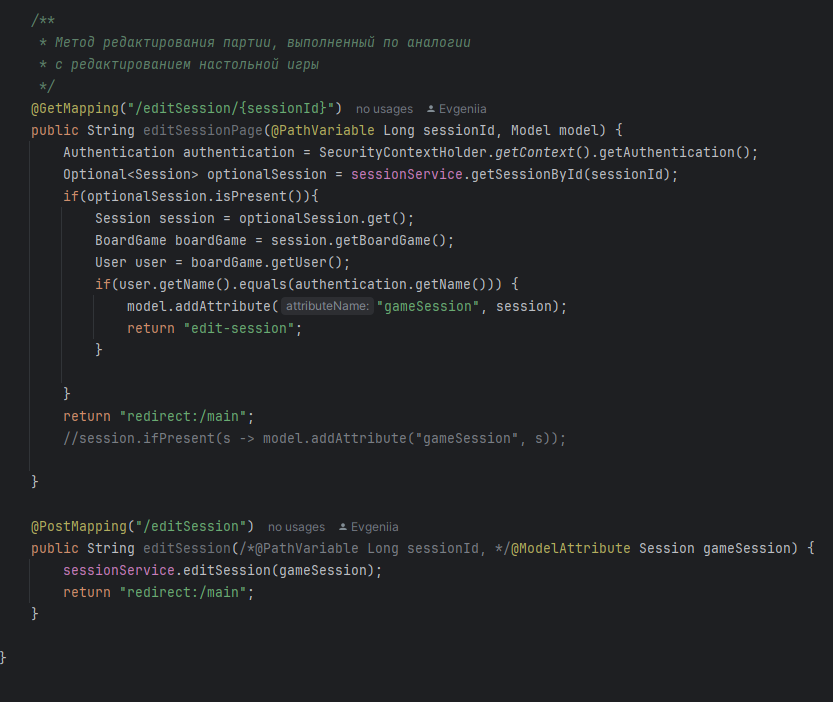


Для редактирования записей о настольных играх необходимо использование двух запросов – GET и POST. Первым шагом является получение информации о существовании запрашиваемого для внесения изменений экземпляра. Если он существует, то по аналогии с представленными выше методами проходит проверка авторизации пользователи, которая в случае успешного прохождения проверки осуществляется переход на HTML-страницу для редактирования записи.

В отличие от @RestController в контроллере, у которого указана аннотация @Controller, для внесения и сохранения изменений применяется метод POST. В editBoardGame пользователь получает возможность отредактировать поля (название, жанр, дата приобретения). Информация   
в полях, которых не коснулись изменения, будут сохранены.

Метод gameSession используется для перехода на страницу с партиями настольной игры. На данном примере можно увидеть отсутствие прямой связи между пользователем и партией, то есть по сути партия относится только   
к настольной игре. Однако из-за привязки настольной игры к пользователю исключается возможность отображения партии у другого пользователя.





На вышеуказанных примерах представлены методы, которые работают с партиями настольных игр. Они выполнены по аналогии с методами контроллера настольных игр. Только для них используется интерфейс сервиса партии (SessionServiceImpl).

В большинстве вышеуказанных методов контроллера используется @PathVariable, служащая для указания переменной в параметре запроса.   
Для передачи переменной в запрос она должна соответствовать значению, указанному между {} в аннотации.

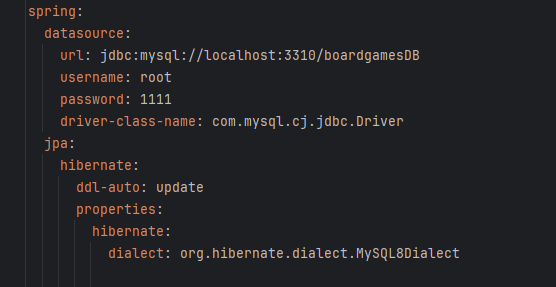
В свою очередь аннотация @RequestBody, которая также используется   
в запросах методов, указывает на необходимость сопоставить тело HTTP-запроса с объектом, а также произвести его автоматическую десериализацию   
в объект Java. Тип, отмеченный данной аннотацией, должен соответствовать JSON, отправленному с контроллера.

Также в контроллерах применяются Model model и аннотация @ModelAttribute. С помощью архитектурного паттерна MVC происходит автоматическое внедрение параметра модели в контроллер. Интерфейс Model используется для передачи данных из контроллера в представление. Он позволяет добавлять атрибуты, которые будут отображаться в шаблонах.   
ModelAttribute необходима для связывания данных из HTTP-запросов   
с объектами модели, а также для передачи данных между контроллером   
и представлением. Таким образом автоматически извлекаются данные   
из формы (или запроса) и заполняются соответствующие поля объекта.

# Глава III. Базы данных

Для данного проекта в качестве СУБД выбрана MySQL, одна из самых популярных реляционных систем управления базами данных. MySQL обладает следующими преимуществами:

1. **Высокая производительность и масштабируемость**: MySQL отлично подходит как для небольших проектов, так и для крупных систем   
   с большими объёмами данных.
2. **Кроссплатформенность**: MySQL можно развёртывать   
   на различных операционных системах, включая Windows, Linux и macOS.
3. **Поддержка сложных запросов и индексов**: MySQL позволяет оптимизировать выборки данных за счёт использования индексов.
4. **Большое сообщество**: Активное сообщество разработчиков   
   и хорошая документация делают MySQL удобным выбором для реализации веб-приложений.



Выше представлена конфигурация подключения СУБД   
к проекту. Для интеграции MySQL с проектом используется Spring Boot, который предоставляет удобный механизм для настройки подключения через файл application.yml.

Для подключения базы данных к проекту необходима настройка конфигурации:

1. Datasource – представляет собой настройки подключения к базе данных:

* jdbc:mysql://localhost:3310/boardgamesDB — URL адрес базы данных.
* localhost – указывает, что база данных запущена на локальном сервере.
* 3310 – порт для подключения к базе данных.
* boardgamesDB – имя базы данных, используемой проектом.

1. Имя пользователя и пароль:

* username: root и password: 1111 задают учетные данные   
  для подключения к базе данных. В данном случае используется пользователь   
  с полными правами (root).

1. Драйвер:

* driver-class-name: com.mysql.cj.jdbc.Driver — указывает   
  на использование драйвера для работы с MySQL. Этот драйвер предоставляет интерфейс для взаимодействия Java-приложения с MySQL.

1. JPA и Hibernate:

* ddl-auto: update – эта настройка указывает Hibernate автоматически создавать, изменять или обновлять таблицы в базе данных в соответствии   
  с классами сущностей.
* dialect: org.hibernate.dialect.MySQL8Dialect: Определяет диалект SQL, который используется для совместимости с MySQL.

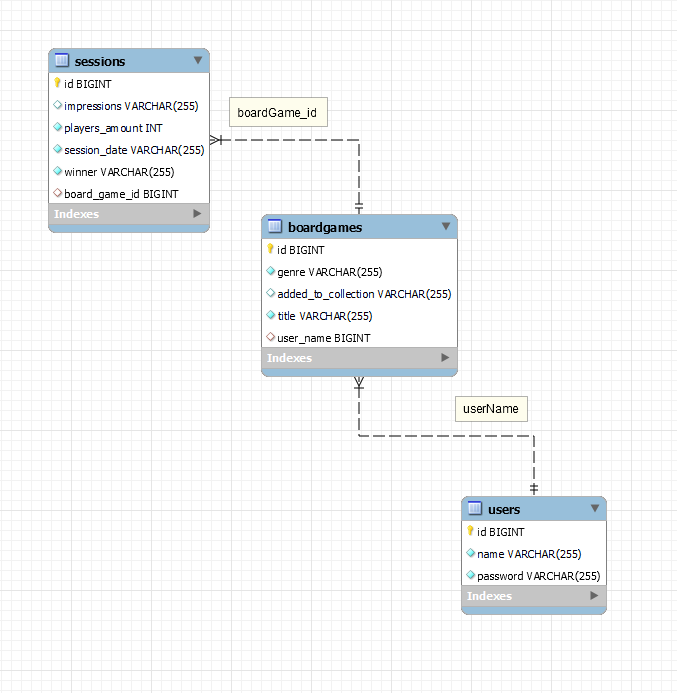
В процессе разработки веб-приложения «Мир Настольных Игр»   
для обеспечения полноты и корректности данных используется три основные таблицы:

users — хранит информацию о пользователях;

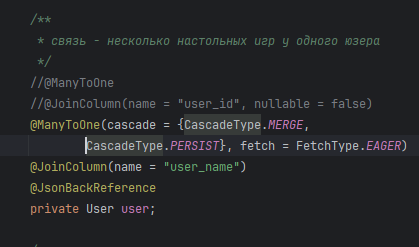
boardgames — хранит данные о настольных играх;

sessions — хранит записи о партиях настольных игр.

Они взаимодействуют следующим образом.



Для установления необходимых связей между классами был добавлен ряд аннотаций.



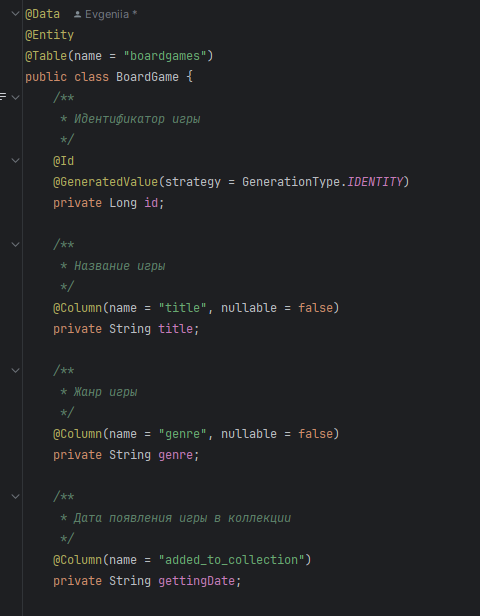
Аннотация @ManyToOne обозначает отношение «многие-к-одному»   
и в классе BoardGame отображает факт того, что один пользователь может обладать несколькими играми.

@JoinColumn используется для определения внешнего ключа, который связывает текущую таблицу с другой таблицей. Это ключевое понятие   
в реляционных базах данных. name: Указывает имя столбца внешнего ключа   
(в данном случае, user\_name).

Если @JoinColumn не указан, JPA автоматически создаст столбец, например, это может быть столбец с ID.

Аннотация @JsonBackReference из библиотеки Jackson используется   
для предотвращения бесконечной рекурсии при сериализации объектов, которые имеют взаимные ссылки (циклические зависимости). Она применяется на стороне «обратной» связи в связях между объектами. Применение этой аннотации позволяет исключить лишние обратные ссылки при отправке запроса на получение данных.

Помимо аннотаций присутствует поле private User user, в котором хранится пользователь.



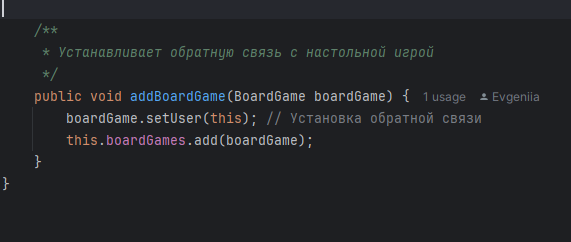
Аннотация @Data из библиотеки Lombok упрощает написание Java-классов, автоматически генерируя код, который разработчики обычно пишут вручную: геттеры, сеттеры, методы toString, equals, hashCode, а также конструкторы.

Аннотация @Entity указывает, что класс является сущностью JPA и будет отображаться на таблицу в базе данных. Каждый экземпляр такого класса соответствует строке в таблице базы данных. По умолчанию имя таблицы совпадает с именем класса, однако с помощью аннотации @Table(name=) можно указать название таблицы по своему выбору.

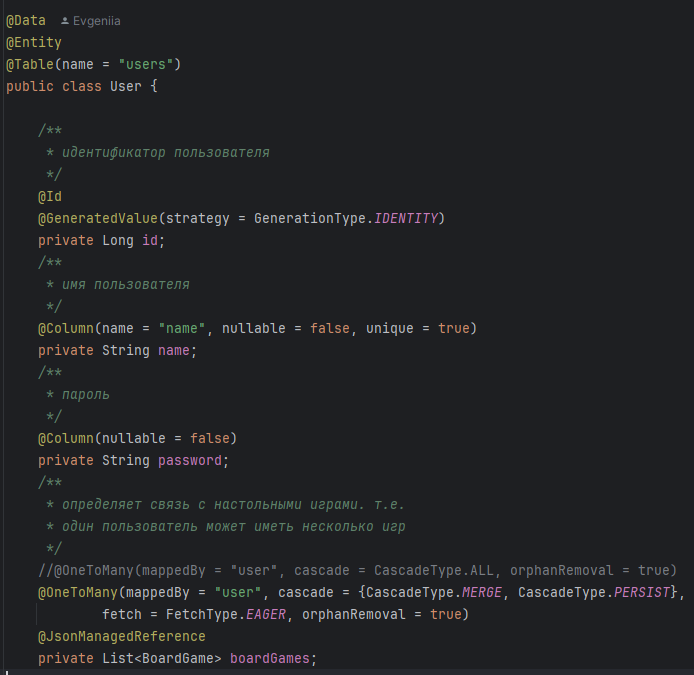
Аннотация @Id используется для обозначения поля, которое будет являться первичным ключом (Primary Key) таблицы.

@GeneratedValue используется совместно с @Id и указывает,   
что значение первичного ключа будет генерироваться автоматически.

Для настройки отображения поля класса как столбца таблицы базы данных используется @Column. Name указывает имя столбца в таблице. Если не указано, используется имя поля. С помощью nullable определяется может   
ли столбец содержать NULL. По умолчанию это true, то есть NULL возможно, но в данном классе название настольной игры и жанр обязательны   
к заполнению.



Помимо аннотаций в классе BoardGame в User также устанавливается обратная связь. С помощью метода addBoardGame экземпляр настольной игры присваивается пользователь, а затем добавляет экземпляр в коллекцию настольных игр пользователя.



@OneToMany описывает связь «один-ко-многим», то есть указывает,   
что один пользователь может иметь несколько настольных игр.

mappedBy: Указывает поле в классе BoardGame, которое отвечает   
за связь. Здесь mappedBy = «user», значит поле user в классе BoardGame описывает обратную связь.

cascade: Определяет поведение каскадных операций (например, сохранение/удаление игр при работе с пользователем).

fetch: Указывает стратегию загрузки. FetchType.EAGER означает,   
что данные загружаются сразу при запросе пользователя.

orphanRemoval: Удаляет «сиротские» записи из базы данных, если они больше не связаны с пользователем.

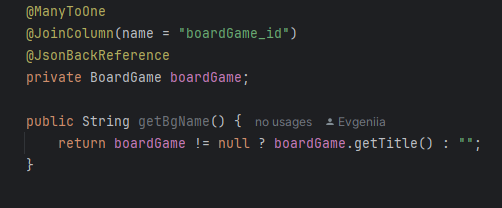
Класс User, как и BoardGame, отмечен аннотациями @Data, @Entity.   
А поле name указано как unique = true, то есть все имена пользователей должны быть уникальными, что важно при их авторизации и аутентификации. Пароль и имя обязательны для заполнения.



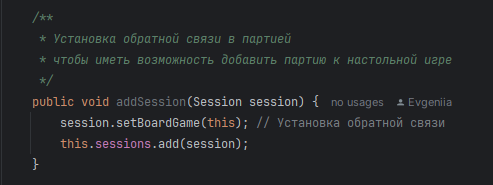
Класс Session описывает структуру таблицы sessions:

* id (PRIMARY KEY): Автоматически генерируемый идентификатор партии.
* sessionDate: Дата проведения партии.
* playersAmount: Количество игроков.
* winner: Имя победителя.
* impressions: Комментарии к партии.

Класс Session разработан по аналогии с классами пользователя   
и настольной игры. Однако в нём производится установка связи с классом BoardGame.



Как и в случае BoardGame с User, одна настольная игра может иметь несколько партий. Для этого применяется аннотация @ManyToOne   
и @JsonBackReference, которая вновь предотвращает бесконечную рекурсию.



И для окончательного закрепления связи между настольной игрой   
и партиями в классе был создан метод addSession. В этом методе идёт присвоение настольной игре партии, а также добавление партии в список.

# Глава IV. Обеспечение безопасности проекта

Spring Security — это мощный и настраиваемый фреймворк   
для обеспечения безопасности Java-приложений. Он предоставляет механизмы аутентификации, авторизации, защиты от атак CSRF, защиты URL и многое другое.

Аутентификация и авторизация являются основными понятиями. Это два фундаментальных процесса, обеспечивающих безопасность в веб-приложениях. Они тесно связаны, но выполняют разные задачи.

Аутентификация — это процесс проверки личности пользователя. Цель аутентификации — убедиться, что пользователь действительно тот, за кого он себя выдаёт.

Основные шаги аутентификации:

* Получение данных пользователя.

Пользователь отправляет свои учетные данные (например, логин   
и пароль) через форму входа.

* Проверка данных.

Приложение проверяет наличие пользователя с указанным именем в базе данных. Если пользователь найден, происходит сравнение введённого пароля   
с хранимым хэшем.

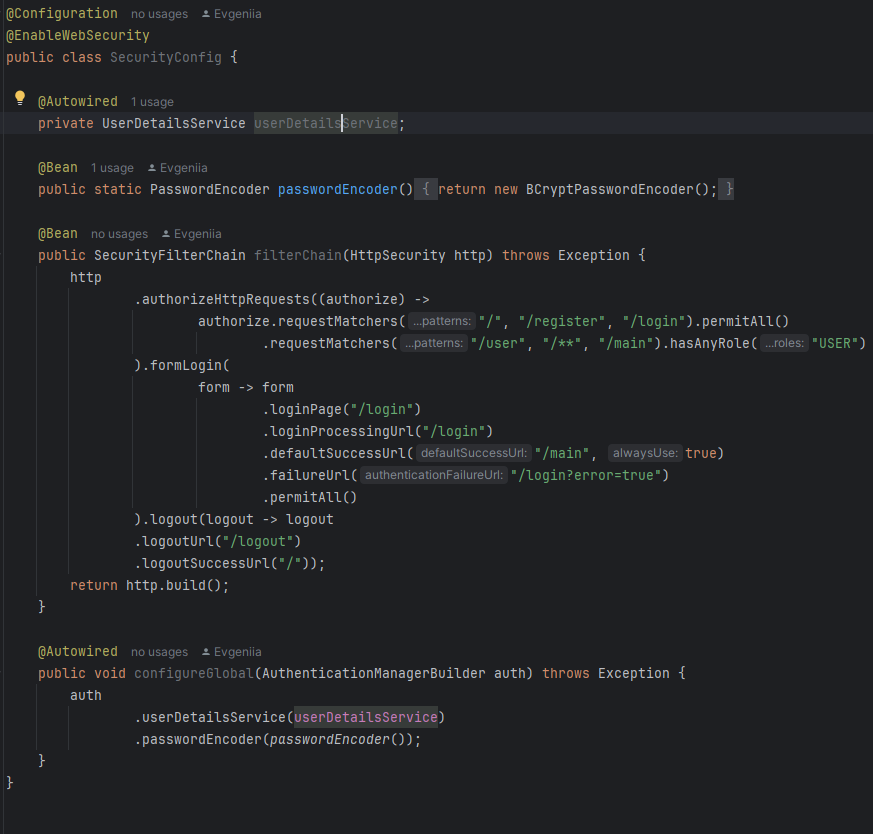
* Успех или ошибка.

При успешной проверке создаётся партия, а пользователь считается аутентифицированным. Если проверка не пройдена, отображается ошибка (например, «Неверный логин или пароль»).

Авторизация — это процесс предоставления прав доступа аутентифицированному пользователю. Цель авторизации — определить, какие ресурсы или действия доступны пользователю в зависимости от его роли.

После аутентификации веб-приложение определяет роль пользователя, на основании которой предоставляет доступ к определённым действия   
и ресурсам.

Для применения SpringSecurity в проект добавлена зависимость   
и разработан класс SecurityConfig.



Для применения SpringSecurity в проект добавлена зависимость   
и разработан класс SecurityConfig.

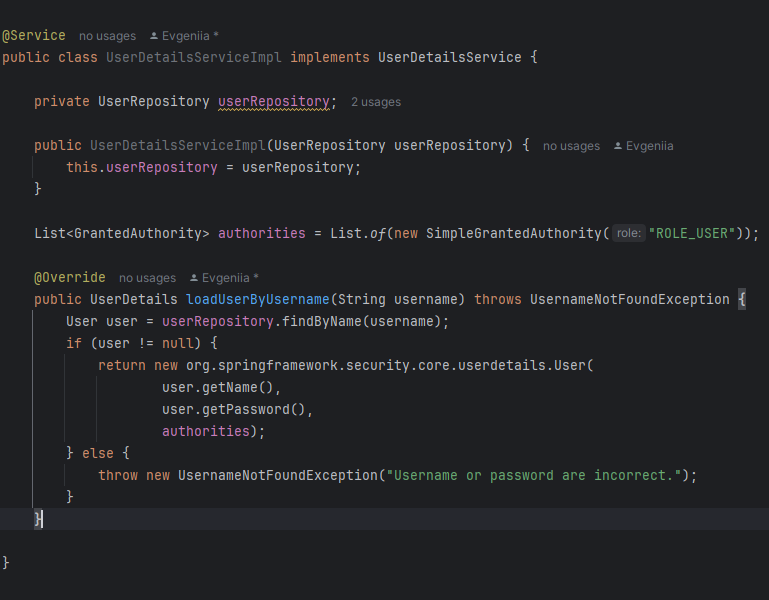
Аннотации @Configuration и @EnableWebSecurity отражают то, что класс является конфигурацией, и в нём осуществляется поддержка безопасности веб-приложения.

Аутентификация осуществляется с помощью UserDetailsService, который используется для загрузки данных пользователя из базы. С помощью BCryptPasswordEncoder осуществляется хэширование паролей.

**Метод SecurityFilterChain инициализирует** фильтр цепочки безопасности, благодаря которому определяются правила доступа к различным URL-адресам.

В веб-приложении «Мир Настольных Игр» страницы /register и /login доступны абсолютно для всех, а остальные страницы уже будут отображаться после авторизации пользователям с ролью USER.

Для конфигурации аутентификации применяется configureGlobal. Он определяет, что userDetailsService будет загружать данные о пользователе   
из базы, а passwordEncoder проверять пароли.



Интерфейс UserDetailsService используется для предоставления данных   
о пользователях. В проекте реализован класс UserDetailsServiceImpl.

Метод loadUserByUsername, представленный в классе, используется   
в Spring Security для загрузки данных пользователя по его имени. В данном случае, предпринимается попытка найти пользователя по имени. В случае, если пользователь найден, предоставляется информация о его имени, пароле   
и проверяется роль. Если такой пользователь отсутствует, выбрасывается исключение UsernameNotFoundException.

Роли задаются через SimpleGrantedAuthority. В данном проекте у всех пользователей роль ROLE\_USER.

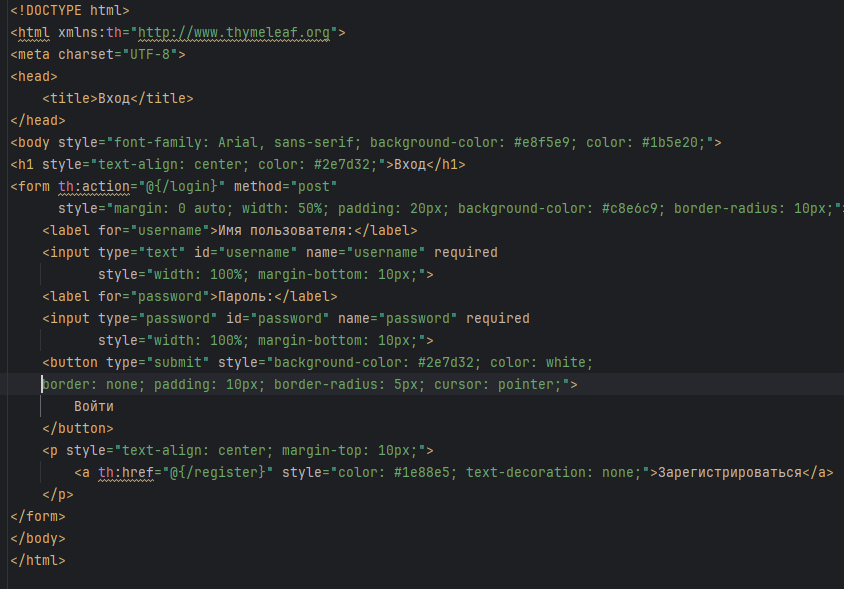
Регистрация пользователей и вход осуществляются через контроллер AuthController с аннотацией @Controller, что означает его принадлежность   
к контроллерам Spring MVC.

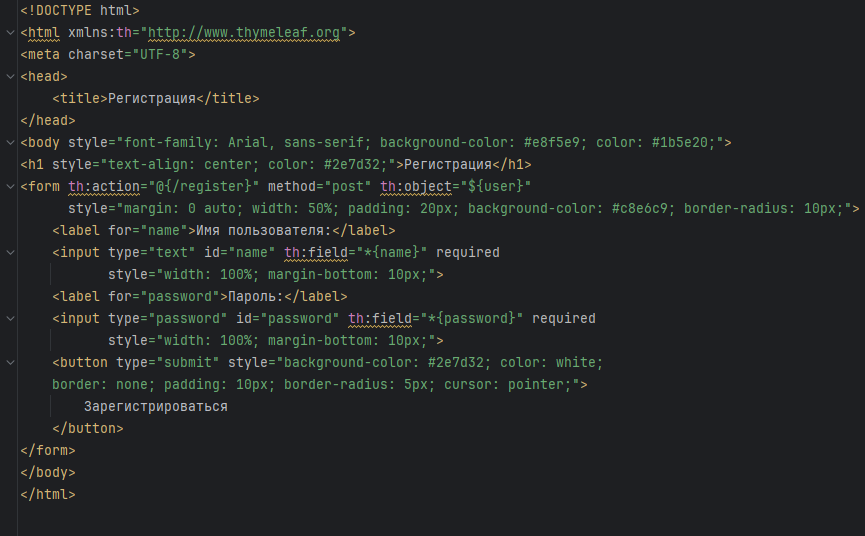
showLoginPage возвращает страницу login, на которой есть форма ввода пользовательских данных для аутентификации, а также позволяет перейти на страницу регистрации.

GET запрос showRegistrationForm добавляет атрибут пользователя   
в модель в представление и возвращается register.

На странице регистрации пользователь вводит свои данные, пароль хэшируется с использованием BCryptPasswordEncoder. Таким образом пользователь добавляется в базу данных и впоследствии может получить доступ к веб-приложению.







На данных примерах представлены страницы регистрации и входа пользователей веб-приложения «Мир Настольных Игр».

На обеих страницах есть заголовки, позволяющие пользователям определить свой функционал. В полях label осуществляется ввод данных (имени и пароля).

Href указывает на то, что пользователь осуществит переход на форму регистрации при нажатии на ссылку. Кнопки отмечаются как button с типом submit, это означает, что при их нажатии происходит отправка формы.

Преимущества использования Spring Security

1. Гибкость:

Лёгкая настройка доступа на основе ролей и URL.

1. Безопасность:

Шифрование паролей, защита от CSRF, встроенные механизмы аутентификации.

1. Удобство интеграции:

Поддержка с различными системами, такими как базы данных, LDAP   
и OAuth2.

1. Масштабируемость:

Подходит как для небольших, так и для крупных приложений.

# Глава V. Docker

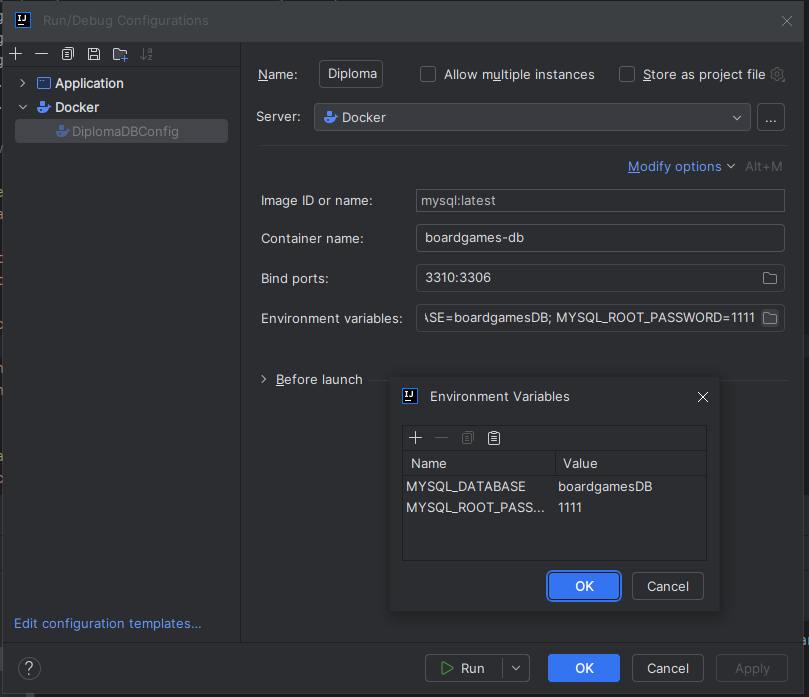
Docker — это платформа для контейнеризации приложений. Контейнеризация позволяет упаковать приложение и его зависимости в единый контейнер, который можно запускать на любом сервере, поддерживающем Docker. Это делает развёртывание приложений более простым, надёжным   
и предсказуемым.

Так как в IntelliJ IDEA имеются встроенные инструменты для работы   
с Docker, его настройка для веб-приложения «Мир Настольных Игр» была осуществлена именно таким образом. Это позволяет управлять контейнерами, образами и сетями прямо из среды разработки, что упрощает настройку   
и тестирование приложений.

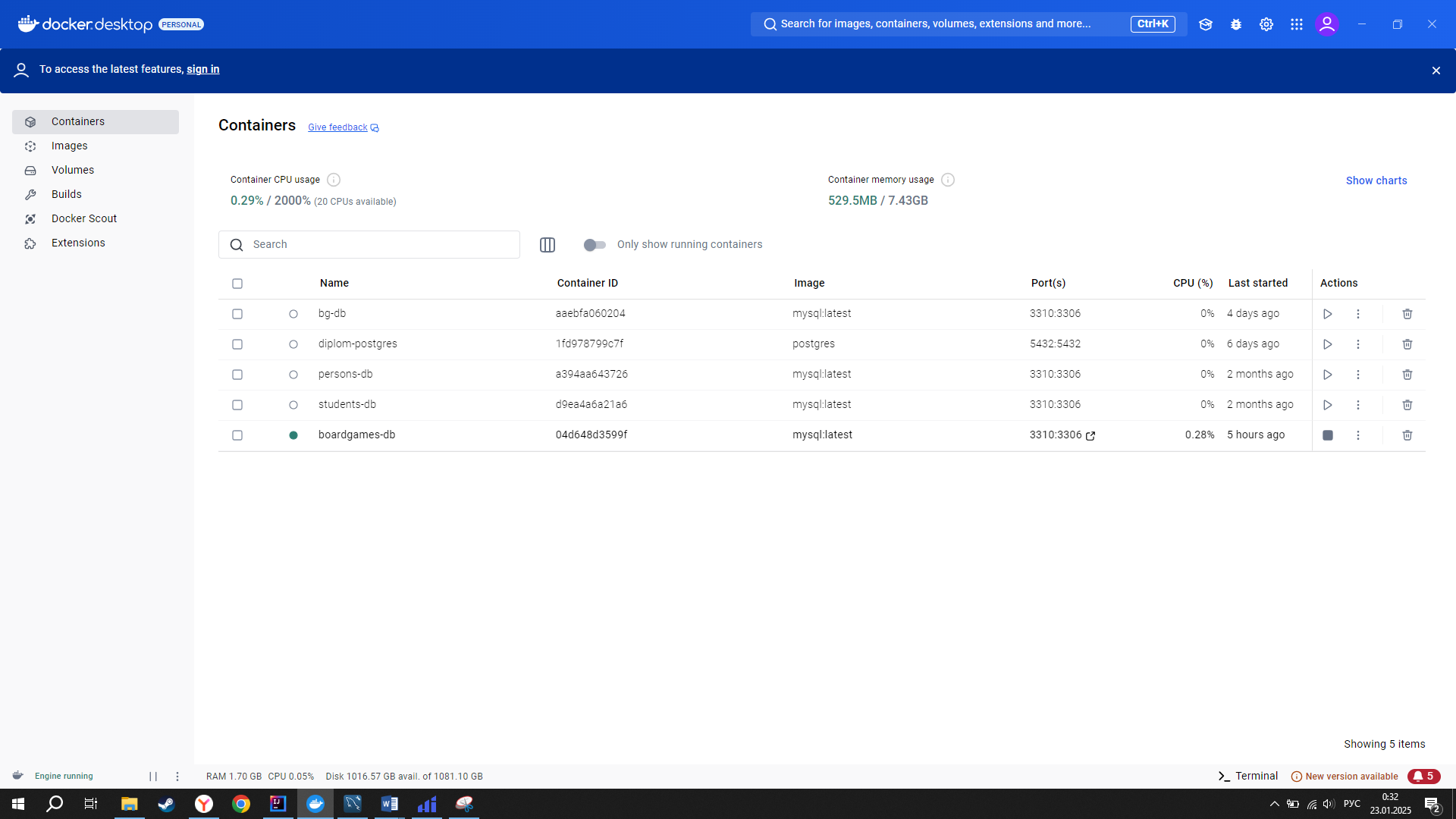
Для работы с Docker необходимо осуществить предварительную подготовку:

1. Установить Docker Desktop.
2. Установить плагин Docker в IntelliJ IDEA.

Настройка Docker в среде разработки выглядит следующим образом:



Для запуска контейнера необходимо, чтобы было запущено приложение Docker Desktop. Запуск контейнера возможен из IntelliJ IDEA через Run.



Преимущества использования Docker через IntelliJ IDEA:

* Удобная интеграция:

Нет необходимости переключаться между терминалом и IDE — всё управление Docker доступно в одном месте.

* Лёгкость отладки:

IntelliJ позволяет запускать контейнеры в режиме отладки.

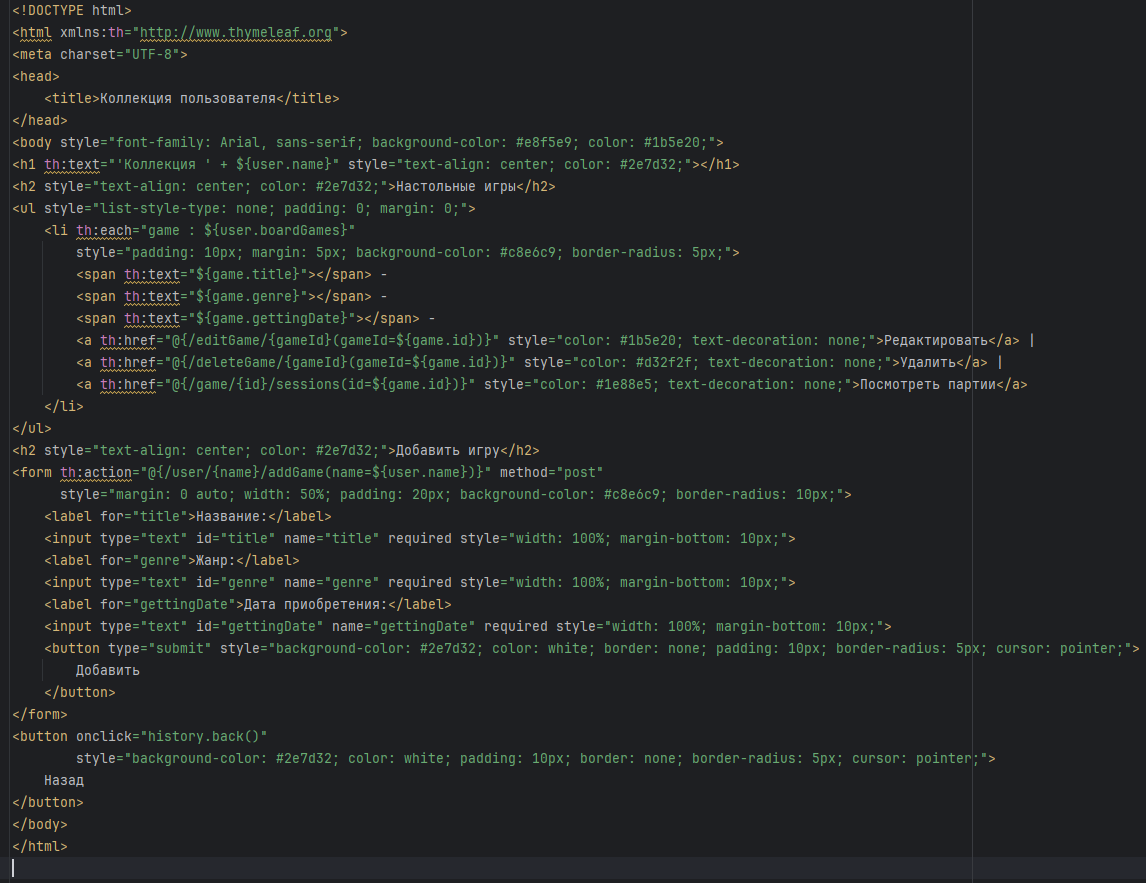
* Управление сервисами:

Можно управлять несколькими контейнерами, сетями и образами через графический интерфейс.

# Глава VI. Применение шаблонизатора Thymeleaf

Thymeleaf — это серверный Java-шаблонизатор, который используется для динамического формирования HTML-страниц. Он предоставляет возможность внедрения данных из серверной части приложения в шаблоны, которые затем отправляются клиенту. Thymeleaf полностью интегрируется   
с фреймворком Spring Boot и позволяет создавать динамические веб-страницы с минимальными усилиями.

В веб-приложение «Мир Настольных Игр» Thymeleaf был внедрён   
для организации навигации между страницами, отображения информации   
о пользователях, настольных играх и партиях, а также для реализации форм   
по добавлению и редактированию экземпляров класса.



В качестве примера будет рассмотрена страница с коллекцией настольных игр пользователя.

В верхней части страницы указана информация о том, что документ относится к типу html, и что используется шаблонизатор Thymeleaf. Тег head содержит в себе заголовок, а body отображает основную информацию   
на странице.

Синтаксис th:text используется для вывода значений модели.

Для итерации по объектам базы данных указывается th:each.

На данной странице пользователь с помощью средств Thymeleaf имеет возможность просмотреть список всех игр, перейти на страницу редактирования или удалить игру, посмотреть партии (помечено <a>).

Также на странице присутствует форма для заполнения данных, которая отвечает за добавление настольных игр, кнопка «назад» для облегчения навигации.

Для более комфортного ознакомления с информацией используется простое оформление.

# Заключение

Разработанный проект веб-приложения «Мир Настольных Игр»   
для управления коллекцией настольных игр успешно реализует поставленные задачи и демонстрирует возможности современных технологий для создания удобных и безопасных инструментов управления данными. Проект охватывает все основные этапы жизненного цикла разработки программного обеспечения: от анализа требований до развёртывания приложения в производственной среде.

**Достижение целей и решение задач:**

Цель проекта заключалась в создании веб-приложения, позволяющего пользователям:

* Управлять коллекцией настольных игр.
* Вести учёт игровых партий, включая их дату, количество игроков, победителя и впечатления.
* Надёжно защищать пользовательские данные, обеспечивая безопасную аутентификацию и авторизацию.

**Для достижения этих целей были выполнены следующие задачи:**

1. Проведён анализ предметной области и определены ключевые требования.

2. Разработана архитектура приложения с использованием паттерна MVC.

3. Реализованы основные компоненты:

* Модель данных для работы с пользователями, играми и партиями.
* Веб-интерфейс для взаимодействия с приложением.
* Логика для обработки данных, включая их сохранение, обновление и удаление.

4. Внедрены механизмы безопасности с помощью Spring Security:

* Аутентификация пользователей с шифрованием паролей.
* Авторизация с разграничением прав доступа.

5. Настроено взаимодействие с реляционной базой данных MySQL, обеспечивающей хранение и целостность данных.

6. Реализован процесс контейнеризации и развёртывания приложения   
с использованием Docker.

**Проблемы и пути их решения.**

В ходе разработки проекта были выявлены следующие проблемы:

1. Обработка больших данных:

При увеличении объёма данных в базе может потребоваться оптимизация запросов и добавление индексов.

2. Оформление интерфейса:

Базовое оформление HTML ограничивает возможности пользовательского интерфейса. Решение: подключение CSS-фреймворков, таких как Bootstrap.

**Возможности дальнейшего развития.**

Проект имеет хороший потенциал для дальнейшего развития. Возможные направления:

1. Добавление новых ролей пользователей:

Введение ролей, таких как ADMIN, для управления другими пользователями и их коллекциями.

2.Интеграция с внешними сервисами:

Подключение к API для получения информации о настольных играх (например, описание и рейтинг).

3. Улучшение интерфейса:

Внедрение CSS-фреймворков для создания более удобного   
и привлекательного интерфейса.

Разработанное веб-приложение является функциональным решением для управления коллекцией настольных игр. Проект демонстрирует современные подходы к разработке и обеспечивает удобный интерфейс, надёжность данных и безопасность. Использование Spring Boot, MySQL и Docker обеспечивает масштабируемость и гибкость приложения, а модульная структура делает его удобным для поддержки и дальнейшего расширения.

Этот проект показал, что современные технологии позволяют быстро и эффективно разрабатывать приложения, соответствующие требованиям пользователей, и обеспечивать их безопасность и удобство

# Список используемой литературы

1. Джош Лонг, Кенни Бастани. Spring в действии. — Москва: Вильямс, 2021.
2. Gavin King и др. Hibernate. Полное руководство. — Санкт-Петербург: Питер, 2019.
3. Марк Хеклер, Spring Boot по-быстрому – Санкт-Петербург, Питер, 2022.
4. Материалы GeekBrains. URL: https://gb.ru.
5. Официальная документация Spring Framework. URL: https://spring.io/projects/spring-framework.
6. Spring Boot Quick Start - www.ivoronline.com, 2021.
7. Thymeleaf Documentation. URL: https://www.thymeleaf.org.