

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное  
образовательное учреждение высшего образования  
«Самарский национальный исследовательский  
университет имени академика С.П. Королева  
(Самарский университет)»

Институт \_\_\_\_\_ информатики и кибернетики  
Кафедра \_\_\_\_\_ программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ**

по дисциплине «Объектная распределенная обработка»

по теме «Игра Пазл»

Выполнил:  
обучающийся группы № 6401-020302D \_\_\_\_\_ Д.О. Колбанов

Проверил:  
*руководитель работы,*  
*доцент кафедры П.С.* \_\_\_\_\_ О.А. Гордеева

Дата защиты \_\_\_\_\_

Оценка \_\_\_\_\_

Самара 2025

## ЗАДАНИЕ

Написать распределенное клиент-серверное приложение, используя указанную технологию распределенной объектной обработки. Приложение реализует игру «Пазл» со следующими правилами:

- 1) игровое поле состоит карточек, которые генерируют поле в зависимости от введенного игроком числа пар;
- 2) каждая ячейка содержит уникальный элемент (Номер), имеющий пару на поле;
- 3) игрок попарно открывает ячейки:
  - если элементы совпадают, ячейки остаются открытыми;
  - если элементы разные, ячейки закрываются через короткий промежуток времени;
- 4) игра завершается, когда все пары найдены.

## РЕФЕРАТ

Пояснительная записка 34 с, рисунков, 11 источников, 1 приложение.

ПАЗЛ, ИГРА, ПРОФИЛЬ, ПОЛЬЗОВАТЕЛЬ, КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, JSP/JSF.

Во время курсовой работы разработаны алгоритмы и соответствующая им программа игры «Пазл». Визуализация происходит на клиентской части системы, а весь игровой процесс – на серверной.

Программа написана на языке Java с использованием среды разработки Eclipse IDE for web developers и функционирует под управлением операционной системы Windows 7 и выше.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	5
1 Описание и анализ предметной области .....	6
1.1 Основные понятия и определения .....	6
1.2 Постановка задачи .....	7
2 Проектирование системы .....	9
2.1 Описание технологии .....	9
2.2 Структурная схема .....	11
2.3 Разработка информационно-логического проекта системы.....	11
2.3.1 Диаграмма вариантов использования .....	12
2.3.2 Диаграмма деятельности.....	13
2.3.3 Диаграмма последовательности .....	14
3 Реализация системы .....	16
3.1 Разработка физической модели БД с использованием PostgreSQL.....	16
3.2 Разработка и описание интерфейса пользователя .....	16
3.3 Диаграмма компонентов .....	20
4 Выводы .....	21
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	22
Приложение А.....	23

## ВВЕДЕНИЕ

В современном мире цифровых технологий компьютерные игры занимают важное место, предоставляя пользователям не только развлечение, но и возможность развивать логическое мышление, внимание и стратегические навыки. Одной из таких игр является "Пазл", которая сочетает в себе простоту правил и увлекательный игровой процесс. Эта игра не только помогает расслабиться, но и стимулирует когнитивные способности, такие как память, концентрация и пространственное мышление [1].

"Пазл" – это классическая логическая игра, в которой игроку необходимо открывать парные ячейки на игровом поле, пытаясь найти совпадения. Если открытые ячейки содержат одинаковые изображения или цвета, они остаются открытыми, в противном случае – закрываются обратно. Игра завершается, когда все ячейки на поле успешно открыты. Простота правил и возможность настройки сложности (за счет изменения размера поля) делают эту игру универсальной и интересной для широкой аудитории [1].

Разработка системы будет производиться по технологии быстрой разработки приложений Rapid Application Development (RAD), которая поддерживается методологией структурного проектирования и включает элементы объектно-ориентированного проектирования и анализа предметной области [2].

При проектировании системы будут использоваться методология Object-Oriented Analysis/Design (ООАП), в основу которой положена объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов, и язык моделирования UML (Unified Modeling Language), который является стандартным инструментом для разработки «чертежей» программного обеспечения.

## 1 Описание и анализ предметной области

Под предметной областью (application domain) принято понимать ту часть реального мира, которая имеет существенное значение или непосредственное отношение к процессу функционирования программы. Другими словами, предметная область включает в себя только те объекты и взаимосвязи между ними, которые необходимы для описания требований и условий решения некоторой задачи [2].

### 1.1 Основные понятия и определения

Игра «Мемори» (Пазл) или другими словами «Найди пару». Мемори – популярная настольная игра для детей дошкольного возраста. Несмотря на простоту конструкции и правил, она несет в себе серьезный потенциал для развития психики ребенка. Это увлекательное и доступное любым возрастам развлечение, которое отлично развивает память и внимание. Игра «мемори» – это карточная настольная игра, состоящая из парных картинок, где основной целью игры является «открытие» как можно большего числа парных карточек, за что игру «мемори» еще называют игра «парочки». Пример игры «Пазл» приведен на рисунке 1.



Рисунок 1 – Пример игры «Пазл»

Правила игры:

- 1) игровое поле состоит из четного числа ячеек, расположенных в виде сетки (например, 4×4 или 6×6 и так далее);
- 2) каждая ячейка содержит уникальный элемент (изображение/цвет), имеющий пару на поле;
- 3) игрок попарно открывает ячейки:
  - если элементы совпадают, ячейки остаются открытыми;
  - если элементы разные, ячейки закрываются через короткий промежуток времени;
- 4) игра завершается, когда все пары найдены.

## 1.2 Постановка задачи

В рамках выполнения курсового проекта требуется разработать распределённое клиент-серверное приложение «Пазл» на базе технологии Java Servlets (JSF), обеспечивающее:

- генерацию игрового поля с четным количеством ячеек;
- интерактивное взаимодействие пользователя с полем через веб-интерфейс;
- проверку совпадения элементов и управление состоянием ячеек;
- сохранение статистики игровых сессий.

В системе будет 1 роль игрок. Пользователь, который занимается поиском пар карточек.

Сервер будет заниматься генерацией игрового поля с уникальными парами элементов, проверять корректность ходов (открытие двух ячеек за ход). Фиксировать время игры и количество попыток. Вести статистику по рейтингу среди пользователей (имя пользователя, количество очков) и личную статистику по игре (время выполнения, количество ходов, начисленные очки, дата выполнения). Проводить аутентификацию и авторизацию пользователей.

На клиентской части для начала игрок должен будет авторизоваться,

введя логин (минимум 4 символа, максимум 30 символов) и пароль (минимум 4 символа, максимум 30 символов), либо зарегистрироваться, если у пользователя еще нет аккаунта, введя логин, пароль, повтор пароля. После авторизации у пользователя будет возможность начать игру. Перед началом игры пользователь должен задать размер поля путем ввода количества пар (минимум 5 максимум 25). На странице игры будет происходить отображение игрового поля с закрытыми и открытыми ячейками, обработка кликов пользователя и отправка запросов на сервер, отображение прогресса (количество оставшихся пар, время).

Функции системы:

- регистрация и авторизация пользователей;
- настройка игрового поля путем указания количества пар в игре;
- генерация случайных элементов (цвет/изображения) для каждой сессии;
- валидация ходов и обновление состояния поля;
- сохранение результатов игры в базе данных;
- формирование рейтинга игроков по времени и количеству ходов.

Функции пользователя:

- создание аккаунта, вход в систему;
- запуск новой игры с выбором количества пар для отгадывания;
- взаимодействие с ячейками через клики;
- просмотр истории игры и личной статистики, просмотр рейтинга среди игроков;
- просмотр справочной информации о системе и о разработчике.



## 2 Проектирование системы

Проектирование является процессом определения структуры, компонентов, интерфейсов и других характеристик системы или ее части. В первую очередь, необходимо определить архитектуру системы. Это включает в себя выбор архитектурного стиля, будь то монолит, микросервисы или клиент-серверная модель, и разработку общей структуры, которая будет служить основой для всех компонентов.

Разрабатываемая система будет веб-приложением, которое передает данные по протоколу HTTP, с двухзвенной клиент-серверной архитектурой и тонким типом клиента.

### 2.1 Описание технологии

JavaServer Faces (JSF) – это стандартная технология Java для создания серверных пользовательских интерфейсов в веб-приложениях. Она была разработана через Java Community Process в рамках JSR-314 и предоставляет набор API для представления UI-компонентов, управления их состоянием, обработки событий, валидации ввода, определения навигации по страницам, а также поддержки интернационализации и доступности [3].

Основные особенности JSF:

- компонентно-ориентированный подход;
- интеграция с JavaServer Pages (JSP);
- поддержка инструментов разработки.

Архитектурные преимущества JSF для системы:

- JSF способствует разделению задач между компонентами, где компоненты определенного слоя работают только с логикой, относящейся к этому слою. Например, компоненты в слое представления работают только с логикой представления, а компоненты в бизнес-слое – только с бизнес-логикой. Это упрощает разработку, тестирование и сопровождение приложения;

– JSF реализует шаблон Model-View-Controller (MVC), что обеспечивает четкое разделение между моделью данных, представлением и контроллером. Это разделение улучшает модульность и упрощает управление сложными приложениями. Пример шаблона MVC приведен на рисунке 2;

– JSF можно расширять и интегрировать с другими технологиями и фреймворками, такими как Facelets для шаблонизации, что обеспечивает гибкость и адаптируемость системы к изменяющимся требованиям.

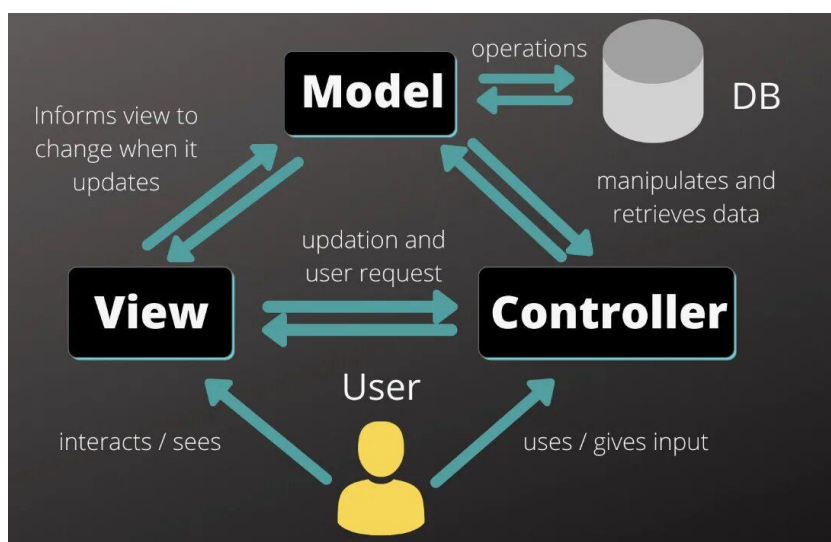


Рисунок 2 – Пример шаблона MVC

## 2.2 Структурная схема

Система (греч. «составленное из частей», «соединение» от «соединяю») – множество элементов, находящихся в отношениях и связях друг с другом, которое образует определённую целостность, единство [4].

Как следует из определения, отличительным (главным свойством) системы является её целостность: комплекс объектов, рассматриваемых в качестве системы, должен обладать общими свойствами и поведением.

Структурная схема – это схема, определяющая основные функциональные части изделия, их назначение и взаимосвязи [5].

На рисунке 3 приведена структурная схема разрабатываемой системы, разделяется клиентскую и серверную часть. Взаимодействие между ними осуществляется по протоколу HTTP стека TCP/IP.

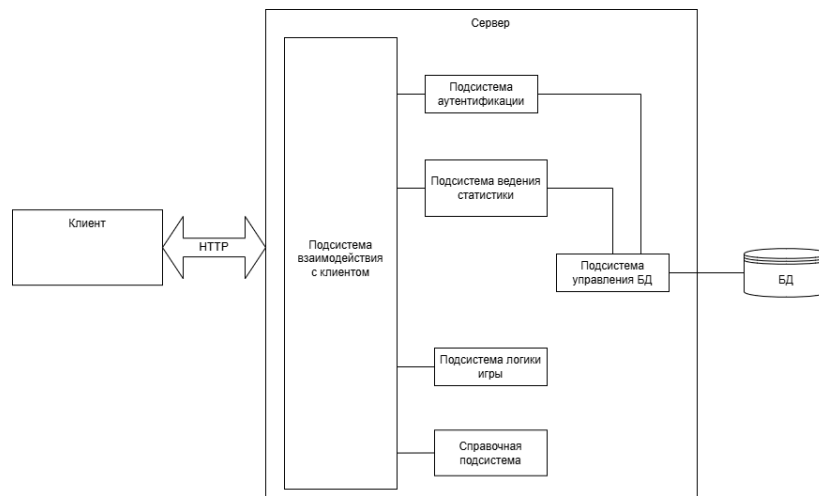


Рисунок 3 – Структурная схема разрабатываемой системы

В состав серверной части входит:

- подсистема взаимодействия с клиентом, которая отвечает за взаимодействие с клиентом;
- подсистема аутентификации, которая отвечает за аутентификацию пользователя в системе;
- подсистема ведения статистики, которая отвечает за работу со статистикой пользователя;
- подсистема логики игры, которая отвечает за логику игры «Быки и коровы»;
- справочная подсистема, которая отвечает за выдачу справочной информации о системе;
- подсистема управления базой данных (БД), которая отвечает за взаимодействие с БД.

БД предназначена для хранения пользовательской информации.

### 2.3 Разработка информационно-логического проекта системы

Цель инфологического проектирования является обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическая модель данных построена по аналогии с естественным языком. Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их атрибуты [6].

### 2.3.1 Диаграмма вариантов использования

Диаграмма вариантов использования представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм. На ней изображаются отношения между актёрами и вариантами использования [7]. На рисунке 4. приведена диаграмма вариантов использования.

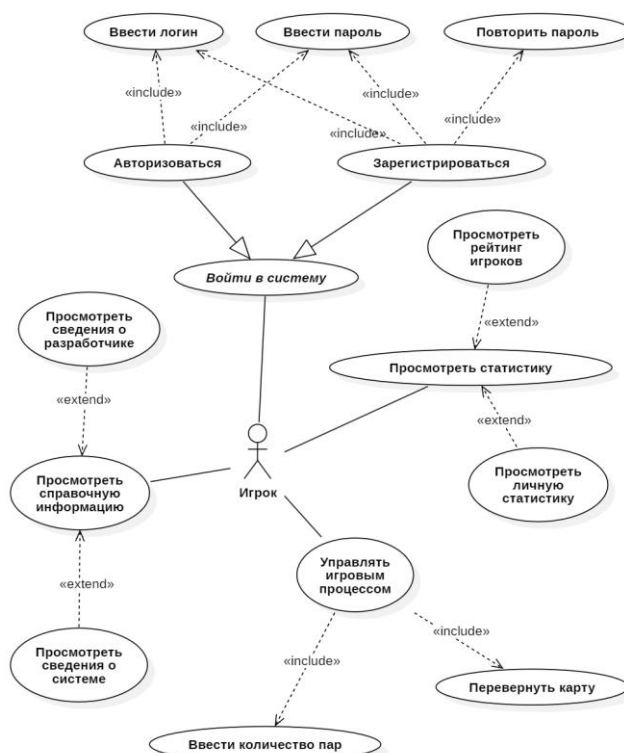


Рисунок 4 – Диаграмма вариантов использования

Игрок имеет возможность войти в систему, для этого ему следует авторизоваться (ввести логин и пароль) или зарегистрироваться (ввести логин, пароль, повторить пароль). Также он может просматривать справочную

информацию о системе и о разработчике. Кроме того, он имеет возможность просмотреть личную статистику и рейтинг среди игроков. Игрок может управлять процессом игры, а именно вводить количество пар для игры, переворачивать карты на игровом поле.

### 2.3.2 Диаграмма деятельности

Диаграмма деятельности – UML-диаграмма, на которой показаны действия, состояния которых описаны на диаграммах состояний [8]. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов – вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

На рисунке 5 приведена диаграмма деятельности системы для всей системы.

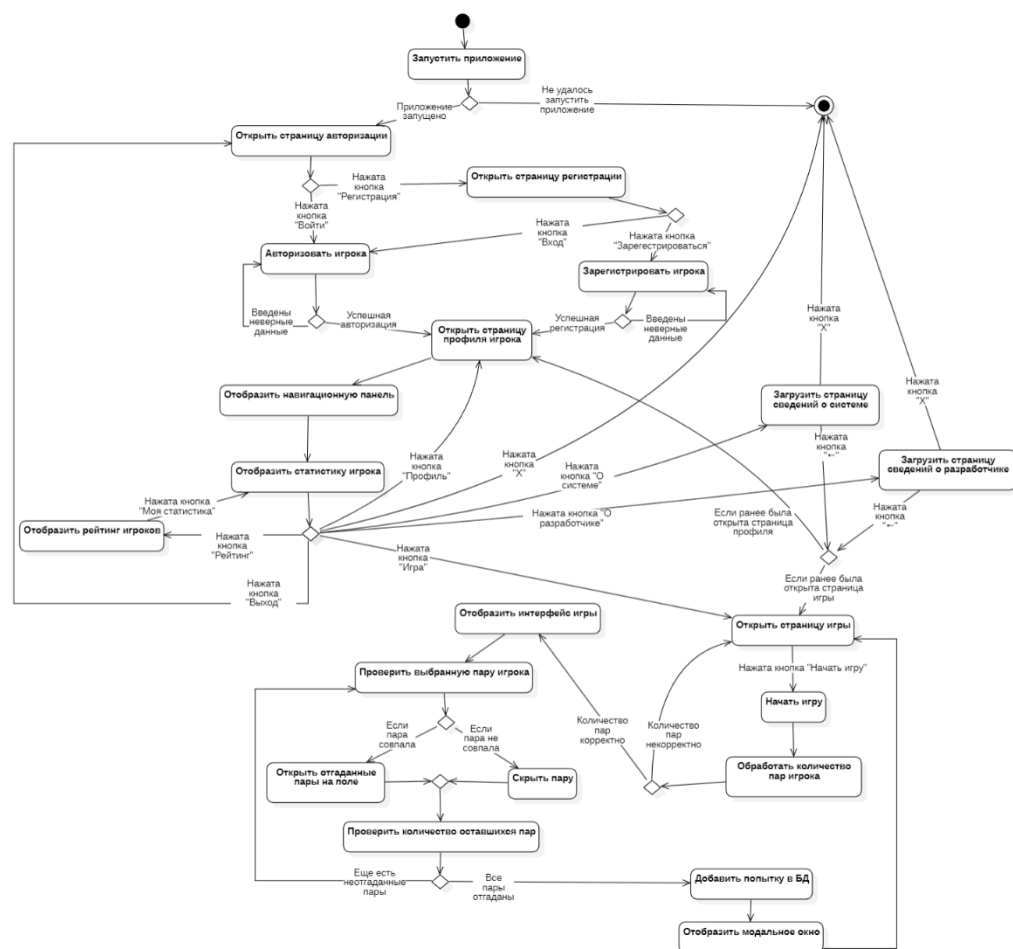


Рисунок 5 – Диаграмма деятельности (Всей системы)

Для работы систему игроку нужно будет запустить приложение в браузере, если все хорошо, то откроется страница авторизации, иначе ничего работать не будет.

После открытия страницы авторизации система предложит игроку авторизоваться либо зарегистрироваться в системе, если у него еще нету аккаунта, если все данные будут введены верно, то система откроет страницу профиля игрока, иначе сообщит о возникших проблемах.

После открытия страницы профиля система отобразит кнопки «Рейтинг» и «Моя статистика» через которые игрок сможет выбирать, какую статистику ему отобразить, а именно рейтинг среди игроков или личная статистика по всем играм. Также система отобразит навигационную панель, через которую игрок сможет перейти на страницу игры, страницы справочной информации или выйти из системы.

При нажатии на кнопку «Игра» система откроет страницу игры, где игрок должен будет ввести, количество пар, а потом нажать кнопку «Начать игру». Система проверит корректность количества, если все хорошо, то отобразит интерфейс игры, иначе сообщит об ошибке. На поле игроку предстоит выбирать пары карточек, а система будет проверять совпадают ли эти пары или нет, если совпадают, то помечать их как отгаданные, иначе скрывать при неудаче. Если оставшихся пар не останется, то игра завершается и система будет выводить модальное окно с результатом.

Также у игрока есть возможность просматривать справочную информацию о системе и о разработчике, нажимая соответствующие кнопки «О системе» и «О разработчике».

При нажатии на кнопку «Выход» система загрузит страницу авторизации, где пользователю снова придется авторизоваться, чтобы воспользоваться возможностями системы.

### 2.3.3 Диаграмма последовательности

Диаграмма последовательности UML (sequence diagram) – такая

диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления. Основные элементы диаграммы последовательности это: обозначения объектов (прямоугольники), вертикальные линии, отображающие течение времени при деятельности объекта, и стрелки, показывающие выполнение действий объектами [9].

На рисунках 6 и 7 приведена диаграмма последовательности для варианта использования «Управлять процессом игры».

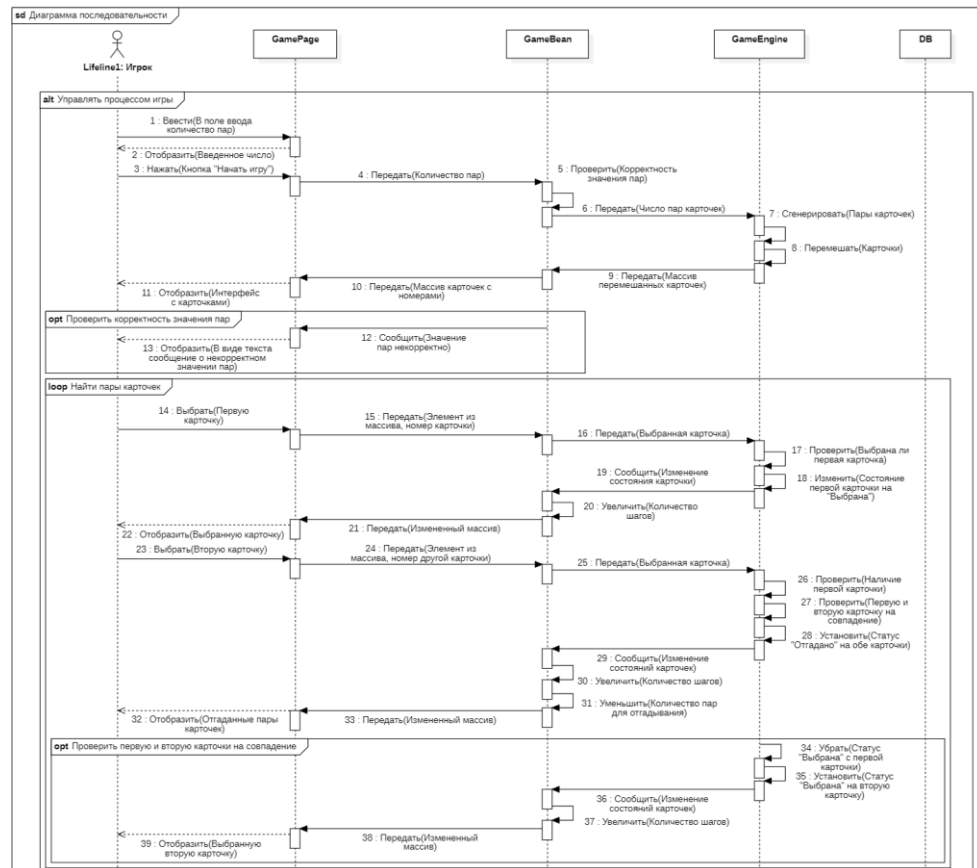


Рисунок 6 – Диаграмма последовательности (Управлять процессом игры 1)

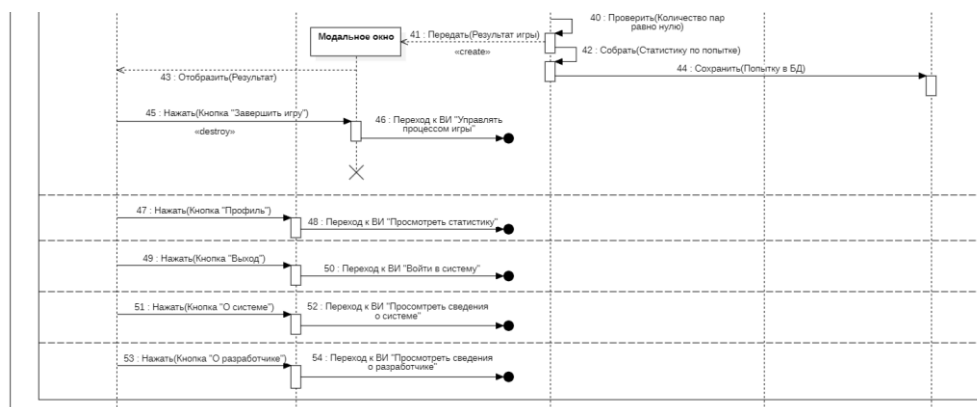


Рисунок 7 – Диаграмма последовательности (Управлять процессом игры 2)

### 3 Реализация системы

#### 3.1 Разработка физической модели БД с использованием PostgreSQL

Физическая модель данных является заключительным шагом в процессе моделирования данных и представляет фактические детали реализации в конкретной СУБД. Моделирование физических данных направлено на оптимизацию производительности операций с базой данных с учетом особенностей и характеристик выбранной СУБД. [10]. Физическая модель представлена на рисунке 7.

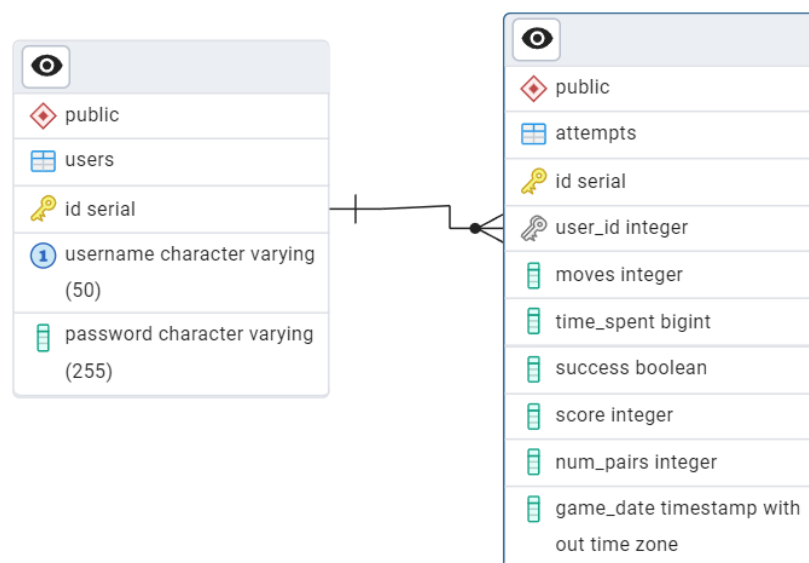


Рисунок 7 – Пример реализации физической модели данных

Физическая модель содержит 2 сущности. Сущность users (игроки) содержит поля id (идентификатор игрока), username (логин игрока), password (пароль). Сущность attempts (попытки) содержит поля id (идентификатор попытки), user\_id (идентификатор игрока), moves (количество ходов), time\_spent (затраченное время на игру), success (исход), score (количество заработанных очков), game\_date (дата игры).

#### 3.2 Разработка и описание интерфейса пользователя

При первом входе в приложение игроку будет предложена возможность авторизовать в системе, если у него уже есть аккаунт, введя свои логин и



пароль, либо пройти регистрацию, нажав кнопку «Регистрация», введя логин, пароль и повтор пароля. После успешного ввода данных система откроет страницу профиля. Формы авторизации и регистрации приведены на рисунке 8.

The image shows two side-by-side forms for user authentication. The left form is for login, with a header containing 'Вход' and 'Регистрация' tabs. It has input fields for 'Логин' (containing 'Dmitry') and 'Пароль' (masked with dots), and a 'Войти' button. The right form is for registration, with a header containing 'Вход' and 'Регистрация' tabs. It has input fields for 'Логин', 'Пароль', and 'Подтверждение пароля', and a 'Зарегистрироваться' button.

Рисунок 8 – Формы авторизации и регистрации

На странице статистики игрок может просмотреть личную статистику, она также открывается при нажатии на кнопку «Моя статистика». Страница с личной статистикой приведена на рисунке 9.

The image shows a user profile page. At the top, there is a navigation bar with buttons: 'Игра', 'Профиль', 'О системе', 'О разработчике', and 'Выход'. Below the navigation bar, it says 'Приветствую, Dmitry!' and has buttons for 'Моя Статистика' and 'Рейтинг'. The main section is titled 'Личная Статистика' and contains a table with 6 columns: 'Количество пар', 'Количество Ходов', 'Время в секундах', 'Успех', 'Счет', and 'Дата Игры'. There are two rows of data in the table.

Количество пар	Количество Ходов	Время в секундах	Успех	Счет	Дата Игры
6	19	19	Да	410	05.03.2025 18:56:43
5	15	14	Да	350	05.03.2025 18:56:15

Рисунок 9 – Страница профиля (Личная статистика)

Чтобы просмотреть рейтинг среди других игроков, который представляет собой таблицу со списком людей и их общим числом очков, необходимо нажать на кнопку «Рейтинг», тогда система отобразит таблицу с рейтингом лучших игроков, которая приведена на рисунке 10.



Игра    Профиль    О системе    О разработке    Выход

Приветствую, Dmitry!

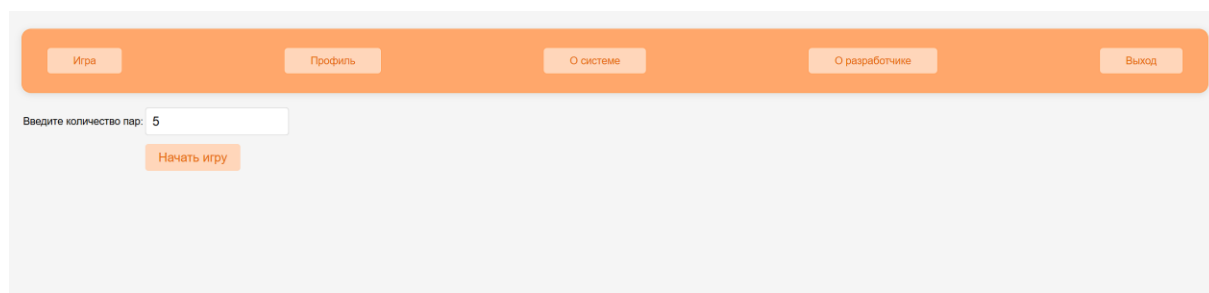
Моя Статистика    Рейтинг

**Рейтинг**

#	Имя Пользователя	Общий Счет
1	Dmitry	760
2	user3	544
3	user1	519
4	user2	467
5	1234	310

Рисунок 10 – Страница профиля (Рейтинг игроков)

У игрока есть возможность переходить по страницам приложения через навигационную панель. В навигационной панели есть кнопка «Играть», при нажатии на которую откроется страница игры, где игроку нужно ввести количество пар, которые нужно сгенерировать для игры. Пример страницы игры с вводом количества пар приведен на рисунке 11.

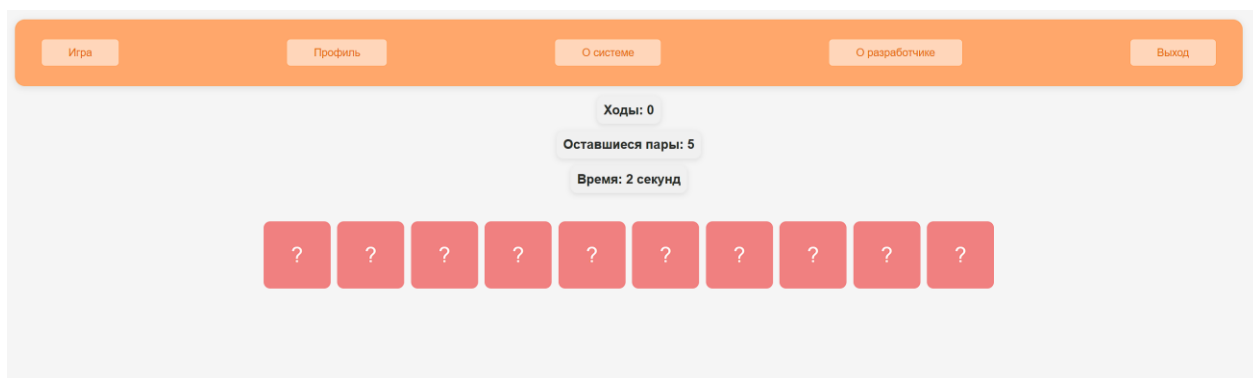


Игра    Профиль    О системе    О разработке    Выход

Введите количество пар:

Рисунок 11 – Страница игры (Ввод количества пар)

После нажатия на кнопку «Начать игру» система проверит корректность значения количества пар и отобразит для игрока интерфейс игры. Здесь игрок будет последовательно открывать карточки и находить те, что совпадают по номерам. Пример страницы игры с интерфейсом игры на рисунке 12.



Игра    Профиль    О системе    О разработке    Выход

Ходы: 0

Оставшиеся пары: 5

Время: 2 секунд

? ? ? ? ? ? ? ? ? ?

Рисунок 12 – Страница игры (Интерфейс игры)

При отгадывании всех пар карточек система отобразит модальное окно с результатами игры, которое изображено на рисунке 13. При нажатии на кнопку «Начать новую игру» система откроет страницу игры с полем ввода пар.



Рисунок 13 – Модальное окно с результатами игры

При нажатии на кнопку «О разработчике» система откроет страницу сведений о разработчике, которая приведена на рисунке 14.

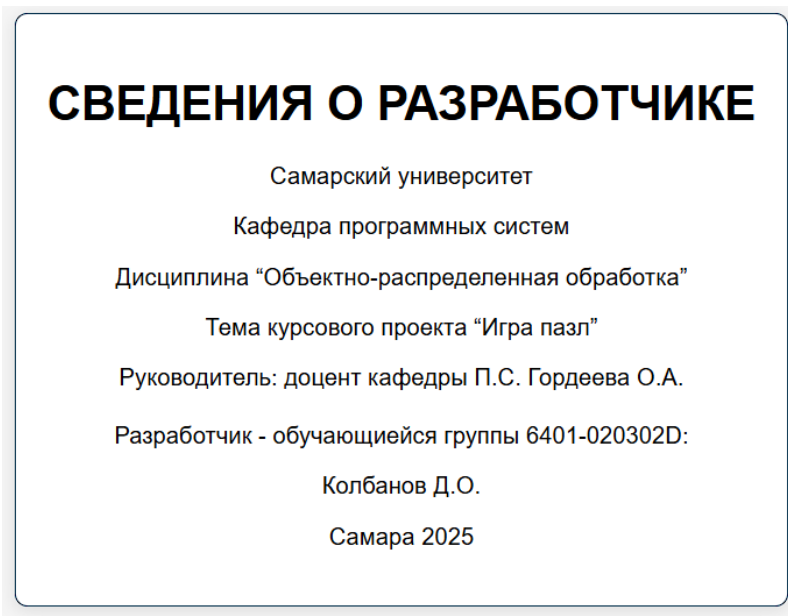


Рисунок 14 – Окно сведений о разработчике

При нажатии на кнопку «О системе» система откроет страницу сведений о системе, которая приведена на рисунке 15.



#### 4 Выводы

В процессе выполнения курсовой работы было разработано распределённое клиент-серверное приложение «Игра «Пазл»» с использованием технологии JSP/JSF.

В первом разделе приведены основные понятия предметной области, рассмотрена история и правила игры «Пазл». На основе проведённого анализа была сформулирована постановка задачи.

Во втором разделе была описана структурная схема системы. Также был разработан информационно-логический проект по методологии UML, в который вошли диаграммы вариантов использования, деятельности всей системы, последовательности для варианта использования «Управлять игровым процессом».

В третьем разделе приведено описание БД, интерфейса пользователя и диаграмма компонентов.

Разработанная система может использоваться пользователями разных возрастов для разнообразия своего досуга.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Найди пару, запомни ячейку. [Электронный ресурс]. URL: <https://www.vdm.ru/products/338/1261/> (дата обращения: 12.02.2025).
- 2 Зеленко Л.С. Технологии программирования и программная инженерия (1 часть) учебное пособие / Л.С. Зеленко. – Самара: изд-во СГАУ, 2006. – 96 с.: ил.
- 3 JavaServer Faces [Электронный ресурс]. URL: [https://pt.wikipedia.org/wiki/JavaServer\\_Faces](https://pt.wikipedia.org/wiki/JavaServer_Faces) (дата обращения: 04.03.2025).
- 4 Функциональная спецификация, Перечень исключительных ситуаций, Спецификация качества - Автоматизированная система построения лабиринта и поиска выхода из него [Электронный ресурс] // Studbooks [сайт]. URL: [https://studbooks.net/2034987/informatika/funktsionalnaya\\_spetsifikatsiya](https://studbooks.net/2034987/informatika/funktsionalnaya_spetsifikatsiya) (дата обращения: 04.03.2025).
- 5 Большой Российский энциклопедический словарь. М.: БРЭ, 2003.
- 6 Информационно-логическое проектирование. [Электронный ресурс] // Studwood.ru: [сайт]. URL: [https://studwood.ru/1884856/informatika/informatsionno\\_logicheskoe\\_proektirovanie](https://studwood.ru/1884856/informatika/informatsionno_logicheskoe_proektirovanie) (дата обращения: 05.03.2025).
- 7 Зеленко Л.С. Методические указания к лабораторному практикуму по дисциплине «Программная инженерия». Самара: СГАУ, 2012. 67 с.
- 8 Диаграмма деятельности [Электронный ресурс] // Википедия [сайт]. URL: [https://ru.wikipedia.org/wiki/Диаграмма\\_деятельности](https://ru.wikipedia.org/wiki/Диаграмма_деятельности) (дата обращения: 05.03.2025).
- 9 Диаграмма деятельности [Электронный ресурс]. URL: [https://ru.wikipedia.org/wiki/Диаграмма\\_деятельности](https://ru.wikipedia.org/wiki/Диаграмма_деятельности) (дата обращения: 06.03.2025).
- 10 Физическое проектирование базы данных [Электронный ресурс]. URL: [https://studopedia.ru/8\\_74489\\_fizicheskoe\\_proektirovanie-bazi-dannih.html](https://studopedia.ru/8_74489_fizicheskoe_proektirovanie-bazi-dannih.html) (дата обращения: 06.03.2025).
- 11 Диаграмма компонентов (component diagram) [Электронный ресурс] // Студопедия: [сайт]. URL: [https://studopedia.ru/6\\_33179\\_diagrammaklassov-class-diagram.html](https://studopedia.ru/6_33179_diagrammaklassov-class-diagram.html) (дата обращения: 06.03.2025).

## ПРИЛОЖЕНИЕ А

### Листинг модулей программы

```
package controllers;

import java.io.Serializable;

public class Card implements Serializable {
    private static final long serialVersionUID = 1L; // Добавьте serialVersionUID

    private int id;
    private boolean isRevealed;
    private boolean isMatched;

    public Card(int id) {
        this.id = id;
        this.isRevealed = false;
        this.isMatched = false;
    }

    public int getId() {
        return id;
    }

    public boolean isRevealed() {
        return isRevealed;
    }

    public void setRevealed(boolean revealed) {
        isRevealed = revealed;
    }

    public boolean isMatched() {
        return isMatched;
    }

    public void setMatched(boolean matched) {
        isMatched = matched;
    }
}

package controllers;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.SessionScoped;
import javax.faces.bean.ViewScoped;

import controllers.UserSessionBean;
import domains.dto.AttemptRequest;
import domains.models.AttemptDao;

import java.io.Serializable;
import java.util.List;

@ManagedBean
@ViewScoped
public class GameBean implements Serializable {

    private GameEngine gameEngine;
    private int numberOfPairs;
    private boolean gameStarted;
    private long startTime; // Время начала игры
    private int moves; // Количество ходов
    private String timeSpent;
    private boolean isStatSave;

    @ManagedProperty(value="#{userSession}")
    UserSessionBean userSession;
```

```

    public UserSessionBean getUserSession() {
        return userSession;
    }

    public void setUserSession(UserSessionBean userSession) {
        this.userSession = userSession;
    }

    public GameBean() {
        gameStarted = false;
        gameEngine = new GameEngine(numberOfPairs);
        isStatSave = false;
    }

    public void resetGame() {
        gameStarted = false;
        gameEngine = new GameEngine(numberOfPairs);
        moves = 0;
        startTime = 0;
        timeSpent = "";
        isStatSave = false;
    }

    public void startNewGame() {
        if (numberOfPairs <= 0) {
            System.out.println("Количество пар должно быть больше нуля.");
            return;
        }
        isStatSave = false;
        gameStarted = true;
        gameEngine = new GameEngine(numberOfPairs); // Создаем новый объект игры с
        новым количеством пар
        this.moves = 0;
        timeSpent = "";
        this.startTime = System.currentTimeMillis();
        System.out.println("Game started with " + numberOfPairs + " pairs.");
    }

    public int getRemainingPairs() {
        return numberOfPairs - gameEngine.getFoundPairs();
    }

    public void selectCard(Card card) {
        gameEngine.selectCard(card);
        moves++; // Увеличиваем количество ходов при каждом выборе карты
    }

    public List<Card> getCards() {
        return gameEngine.getCards();
    }

    public boolean isGameComplete() {
        if (gameEngine.isGameComplete() && gameStarted) {
            if (!isStatSave) {
                saveGameStatistics();
                isStatSave = !isStatSave;
            }

            return true;
        }
        return false;
    }

    private void saveGameStatistics() {
        long endTime = System.currentTimeMillis();
        timeSpent = String.valueOf((endTime - startTime) / 1000); // Вычисляем
        затраченное время в миллисекундах
    }

```



```

        // Создаем объект AttemptRequest для сохранения в базе данных
        AttemptRequest attemptRequest = new AttemptRequest(
            userSession.getUser().getId(), // Получаем ID пользователя из сессии
            moves,
            timeSpent, // Переводим миллисекунды в секунды
            true, // Успех (можно изменить в зависимости от логики игры)
            calculateScore(), // Метод для расчета баллов
            numberOfPairs
        );

        // Сохраняем попытку через DAO
        AttemptDao attemptDao = new AttemptDao();
        attemptDao.addAttempt(attemptRequest);
    }

    private int calculateScore() {
        // Пример простой логики для расчета баллов
        return Math.max((numberOfPairs * 100) - (moves * 10), 0); // Чем меньше ходов,
        тем выше балл
    }

    public int getCountMove() {
        return moves;
    }

    public int getNumberOfPairs() {
        return numberOfPairs;
    }

    public void setNumberOfPairs(int numberOfPairs) {
        this.numberOfPairs = numberOfPairs;
    }

    public boolean isGameStarted() {
        return gameStarted;
    }

    public void setGameStarted(boolean gameStarted) {
        this.gameStarted = gameStarted;
    }

    public String getTimeSpent() {
        return timeSpent;
    }

    public void setTimeSpent(String timeSpent) {
        this.timeSpent = timeSpent;
    }
}

package controllers;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class GameEngine implements Serializable {
    private static final long serialVersionUID = 1L; // Добавьте serialVersionUID

    private int gridSize;
    private List<Card> cards;
    private Card firstSelectedCard;
    private Card secondSelectedCard;
    private boolean isCheckingMatch;
    private int pairsFound;

    public GameEngine(int numberOfPairs) {
        this.gridSize = numberOfPairs * 2;
    }
}

```

```

        this.cards = generateCards(numberOfPairs);
        this.pairsFound = 0;
        this.isCheckingMatch = false;
    }

    public int getFoundPairs() {
        return pairsFound;
    }

    private List<Card> generateCards(int numberOfPairs) {
        List<Card> cards = new ArrayList<>();
        for (int i = 0; i < numberOfPairs; i++) {
            Card card1 = new Card(i);
            Card card2 = new Card(i);
            cards.add(card1);
            cards.add(card2);
        }
        Collections.shuffle(cards);
        return cards;
    }

    public void selectCard(Card selectedCard) {
        if (isCheckingMatch || selectedCard.isMatched() ||
selectedCard.equals(firstSelectedCard)) {
            return;
        }

        selectedCard.setRevealed(true);

        if (firstSelectedCard == null) {
            firstSelectedCard = selectedCard;
        } else {
            secondSelectedCard = selectedCard;
            isCheckingMatch = true;
            checkMatch();
        }
    }

    private void checkMatch() {
        if (firstSelectedCard.getId() == secondSelectedCard.getId()) {
            firstSelectedCard.setMatched(true);
            secondSelectedCard.setMatched(true);
            pairsFound++;
            firstSelectedCard = null;
            secondSelectedCard = null;
            System.out.println("Match found!");
        } else {
            System.out.println("No match, hiding cards...");
            firstSelectedCard.setRevealed(false);
            secondSelectedCard.setRevealed(true);
            firstSelectedCard = secondSelectedCard;
        }
        isCheckingMatch = false;
    }

    public boolean isGameComplete() {
        return pairsFound == gridSize / 2;
    }

    public List<Card> getCards() {
        return cards;
    }
}
package controllers;

import java.util.List;

import javax.annotation.PostConstruct;

```

```

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;

import domains.dto.AttemptResponse;
import domains.dto.UserScoreResponse;
import domains.models.AttemptDao;

import javax.faces.bean.ManagedProperty;

@ManagedBean
@ViewScoped
public class ProfileBean {
    private AttemptDao attemptDao = new AttemptDao();
    private List<UserScoreResponse> userScores; // Для рейтинга
    private List<AttemptResponse> userAttempts; // Для личной статистики
    private int currentUserId; // ID текущего пользователя
    private boolean showStatistic1 = true;

    @ManagedProperty(value="#{userSession}")
    private UserSessionBean userSession;

    public UserSessionBean getUserSession() {
        return userSession;
    }

    public void setUserSession(UserSessionBean userSession) {
        this.userSession = userSession;
    }

    @PostConstruct
    public void init() {
        this.currentUserId = userSession.getUser().getId();
        loadUserScores();
        loadUserAttempts();
    }

    public void loadUserScores() {
        showStatistic1 = false;
        userScores = attemptDao.getUserScores();
    }

    public void loadUserAttempts() {
        showStatistic1 = true;
        userAttempts = attemptDao.getAttemptsByUserId(currentUserId);
    }

    public List<UserScoreResponse> getUserScores() {
        return userScores;
    }

    public List<AttemptResponse> getUserAttempts() {
        return userAttempts;
    }

    // Метод для переключения на личную статистику
    public void showUserAttempts() {
        loadUserAttempts();
    }

    public boolean getShowStatistic1() {
        return showStatistic1;
    }

    public void setShowStatistic1(boolean showStatistic1) {
        this.showStatistic1 = showStatistic1;
    }
}

```

```

}
package controllers;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;

import domains.dto.CustomUser;
import services.AuthService;
import utils.config;

@ManagedBean(name = "userSession") // Имя бина
@SessionScoped
public class UserSessionBean {
    private CustomUser user = new CustomUser(); // Инициализация пользователя
    private boolean loginFormVisible = true;
    private boolean registerFormVisible = false;

    public String login() {
        AuthService authService = new AuthService();
        CustomUser authenticatedUser = authService.authenticate(user.getUsername(),
user.getPassword());

        if (authenticatedUser != null) {
            this.user = authenticatedUser ; // Сохраняем аутентифицированного
пользователя
            System.out.println("Авторизация успешна");
            return config.PROFILE_PAGE_FILE_PATH; // Перенаправление на домашнюю
страницу
        }
        FacesContext.getCurrentInstance().addMessage(null, new FacesMessage("Неверный
логин или пароль."));
        System.out.println("Неверный логин или пароль.");
        return null; // Возвращаем null для отображения ошибки
    }

    public String logout() {
        System.out.println("Logout метод вызван"); // Добавьте этот вывод
        if (user != null) {
            new AuthService().logout(user);
            user = new CustomUser(); // Завершение сессии
        }
        System.out.println("Пока пока, я выхожу...");
        return config.AUTH_PAGE_FILE_PATH; // Перенаправление на страницу входа
    }

    public String register() {
        AuthService authService = new AuthService();

        // Проверка длины логина и пароля
        if (user.getUsername().length() < 4 || user.getUsername().length() > 30) {
            FacesContext.getCurrentInstance().addMessage(null, new FacesMessage("Логин
должен содержать от 4 до 30 символов."));
            return null; // Возвращаем null для отображения ошибки
        }
        if (user.getPassword().length() < 4 || user.getPassword().length() > 30) {
            FacesContext.getCurrentInstance().addMessage(null, new FacesMessage("Пароль
должен содержать от 4 до 30 символов."));
            return null; // Возвращаем null для отображения ошибки
        }

        // Проверка на совпадение паролей
        if (!user.getPassword().equals(user.getConfirmPassword())) {
            FacesContext.getCurrentInstance().addMessage(null, new FacesMessage("Пароли
не совпадают."));
            return null; // Возвращаем null для отображения ошибки
        }
    }
}

```

```

        // Проверка на существование пользователя
        CustomUser authenticatedUser = authService.register(user.getUsername(),
user.getPassword());
        if (authenticatedUser != null) {
            System.out.println("Регистрация успешна");
            this.user = authenticatedUser ;
            return config.PROFILE_PAGE_FILE_PATH; // Перенаправление на домашнюю
страницу
        }

        // Если пользователь с таким именем уже существует
        FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage("Пользователь с таким именем уже существует."));
        return null; // Возвращаем null для отображения ошибки
    }

    public void showLoginForm() {
        this.loginFormVisible = true;
        this.registerFormVisible = false;
    }

    public void showRegisterForm() {
        this.loginFormVisible = false;
        this.registerFormVisible = true;
    }

    public boolean isLoggedIn() {
        return user != null && user.isLoggedIn();
    }

    public CustomUser getUser () {
        return user;
    }

    // Геттеры для username и password
    public String getUsername() {
        return user.getUsername();
    }

    public void setUsername(String username) {
        user.setUsername(username);
    }

    public String getPassword() {
        return user.getPassword();
    }

    public void setPassword(String password) {
        user.setPassword(password);
    }

    public String getConfirmPassword() {
        return user.getConfirmPassword();
    }
    public void setConfirmPassword(String confirmPassword) {
        user.setConfirmPassword(confirmPassword);
    }

    public boolean isLoginFormVisible() {
        return loginFormVisible;
    }

    public void setLoginFormVisible(boolean loginFormVisible) {
        this.loginFormVisible = loginFormVisible;
    }

```

```

public boolean isRegisterFormVisible() {
    return registerFormVisible;
}

public void setRegisterFormVisible(boolean registerFormVisible) {
    this.registerFormVisible = registerFormVisible;
}
}
package domains;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;

import utils.DatabaseUtil;

@WebListener
public class AppContextListener implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        DatabaseUtil.initializeDatabase();
        System.out.println("База данных инициализирована при старте приложения.");
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        // Здесь можно закрыть соединения или выполнить другие действия при остановке
        приложения
    }
}
package domains.dto;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class Attempt {
    private int userId;           // Идентификатор пользователя
    private int moves;            // Количество ходов
    private String timeSpent;     // Время, затраченное на попытку
    private boolean success;      // Успех попытки
    private int score;           // Балл, полученный за попытку
    private int numPairs;
    private LocalDateTime gameDate; // Дата и время игры

    // Формат для отображения даты
    private static final DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss");

    public Attempt() {
    }

    public Attempt(int userId, int moves, String timeSpent, boolean success, int score,
int numPairs, LocalDateTime gameDate) {
        this.userId = userId;
        this.moves = moves;
        this.timeSpent = timeSpent;
        this.success = success;
        this.score = score;
        this.numPairs = numPairs;
        this.gameDate = gameDate;
    }

    public String getResult() {
        if (success) {
            return "Да";
        }
    }
}

```

```

        else {
            return "Her";
        }
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getMoves() {
        return moves;
    }

    public void setMoves(int moves) {
        this.moves = moves;
    }

    public String getTimeSpent() {
        return timeSpent;
    }

    public void setTimeSpent(String timeSpent) {
        this.timeSpent = timeSpent;
    }

    public boolean isSuccess() {
        return success;
    }

    public void setSuccess(boolean success) {
        this.success = success;
    }

    public int getScore() {
        return score;
    }

    public void setScore(int score) {
        this.score = score;
    }

    public LocalDateTime getGameDate() {
        return gameDate;
    }

    public void setGameDate(LocalDateTime gameDate) {
        this.gameDate = gameDate;
    }

    // Метод для получения отформатированной даты
    public String getFormattedGameDate() {
        return gameDate.format(formatter);
    }

    public int getNumPairs() {
        return numPairs;
    }

    public void setNumPairs(int numPairs) {
        this.numPairs = numPairs;
    }
}

package filters;

import java.io.IOException;

```

```

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import controllers.UserSessionBean;

@WebFilter(urlPatterns = "/faces/protected/*") // Укажите защищенные URL
public class AuthFilter implements Filter {
    private static final long serialVersionUID = 1L;

    @Override
    public void init(FilterConfig fConfig) throws ServletException {
        System.out.println("Инициализируем филтр!");
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
        throws IOException, ServletException {

        HttpServletRequest httpRequest = (HttpServletRequest) request;
        HttpServletResponse httpResponse = (HttpServletResponse) response;
        HttpSession session = httpRequest.getSession(false);

        // Проверьте, авторизован ли пользователь
        if (session != null && session.getAttribute("userSession") != null) {
            UserSessionBean userSession = (UserSessionBean)
session.getAttribute("userSession");
            if (userSession.isLoggedIn()) {
                // Пользователь авторизован, продолжайте
                chain.doFilter(request, response);
                return;
            }
        }

        // Пользователь не авторизован, перенаправляем на страницу логина
        System.out.println(httpRequest.getContextPath() + "/faces/pages/login.xhtml");
        httpResponse.sendRedirect(httpRequest.getContextPath() +
"/faces/pages/login.xhtml");
    }
}
package services;

import domains.dto.CustomUser;
import domains.dto.UserRequest;
import domains.dto.UserResponse;
import domains.models.UserDao;

public class AuthService {
    private UserDao userDao = new UserDao();

    public CustomUser authenticate(String username, String password) {
        UserResponse user = userDao.findByUsername(username);
        if (user == null) {
            return null;
        }
        CustomUser advanceUser = new CustomUser(user);
        if (advanceUser.getPassword().equals(password)) {
            advanceUser.setLoggedIn(true);

```



```

        return advanceUser ;
    }
    return null;
}

public CustomUser register(String username, String password) {
    // Проверка, существует ли пользователь с таким же именем
    if (userDao.findByUsername(username) != null) {
        System.out.println("Пользователь с таким именем уже существует");
        return null; // Пользователь уже существует
    }
    UserRequest user = new UserRequest(username, password);
    userDao.save(user);
    CustomUser advanceUser = new CustomUser(userDao.findByUsername(user.getUsername()));
    advanceUser.setLoggedIn(true);
    return advanceUser; // Регистрация успешна
}

public void logout(CustomUser user) {
    user.setLoggedIn(false);
}
}

package domains.dto;

public class UserRequest extends User {
    public UserRequest(String username, String password) {
        super(username, password);
    }
}
}

```