Geekbrains

## Разработка веб-приложения для интерактивного тестирования знаний основ языка Java (Quiz Game)

Программа:

Специализация

Рассолова К.Э.

Апатиты

2025

**Содержание**

Введение

Глава 1. Теоретическая часть

1.1 Технологии и подходы для разработки веб-приложений

1.2 Анализ существующих решений

Глава 2. Практическая часть

2.1 Архитектура приложения

2.2 Реализация функционала

2.3 Интерфейс пользователя

2.4 Пример использования

Заключение

Список используемой литературы

Приложения

**Введение**

Современные технологии обучения активно развиваются, и одним из ключевых инструментов в образовательной среде становятся веб-приложения. Проект "Quiz Game" направлен на создание удобного и функционального решения для проверки знаний пользователей в интерактивной форме.

**Обоснование темы проекта**

Образовательные технологии играют важную роль в формировании знаний и навыков у учащихся, что подтверждается исследованиями: более 70% студентов и преподавателей используют цифровые платформы для обучения. Однако многие из них сталкиваются с ограничениями в функциональности или сложностью настройки существующих систем тестирования. "Quiz Game" стремится решить эти проблемы, предлагая интуитивно понятный интерфейс, расширенные функции аналитики и акцент на ассоциациях.

**Цель проекта**

Целью разработки является создание веб-приложения, которое обеспечит:

* Простую настройку тестов и управление вопросами.
* Поддержку адаптивного тестирования с анализом результатов.
* Визуализацию прогресса и аналитические отчёты для пользователей.

**Проблематика**

Существующие системы тестирования часто страдают от следующих недостатков:

* Ограниченные возможности для создания тестов и анализа результатов.
* Отсутствие гибкости в настройке структуры вопросов.
* Недостаточная поддержка пользовательского интерфейса на разных устройствах. Проект "Quiz Game" решает эти проблемы, предлагая современное решение с использованием передовых технологий.

**Задачи проекта**

* Изучить существующие платформы тестирования и их недостатки.
* Спроектировать архитектуру приложения с использованием Spring Boot и MySQL.
* Реализовать REST API для управления вопросами и результатами тестов.
* Разработать пользовательский интерфейс с адаптивным дизайном.
* Провести тестирование приложения для проверки его надёжности и функциональности.

**Инструменты и технологии**

Для реализации проекта выбраны следующие технологии:

* **Java и Spring Boot**: для создания серверной логики и REST API.
* **MySQL**: для хранения данных о пользователях, тестах и результатах.
* **HTML/CSS/JavaScript**: для реализации клиентской части приложения.
* **Docker**: для контейнеризации приложения и упрощения развёртывания.

**Состав команды**

Проект выполнен самостоятельно, с распределением ролей:

* **Разработчик**: проектирование и реализация серверной части.
* **Тестировщик**: проверка функциональности и устранение ошибок.
* **Дизайнер**: создание интерфейса и визуальных элементов приложения.

**Глава 1. Теоретическая часть**

**1.1 Технологии и подходы для разработки веб-приложений**

**Обзор Java и его применения в веб-разработке**

Java является одним из самых популярных языков программирования для разработки веб-приложений благодаря своей платформонезависимости и богатой экосистеме библиотек. Основные преимущества Java в веб-разработке:

* **Платформонезависимость**: код, написанный на Java, может выполняться на любой системе с установленной JVM.
* **Высокая производительность**: благодаря JIT-компиляции Java обеспечивает производительность, сравнимую с нативными приложениями.
* **Безопасность**: встроенные механизмы безопасности делают Java надёжным выбором для разработки веб-приложений.
* **Масштабируемость**: Java поддерживает разработку масштабируемых приложений благодаря фреймворкам, таким как Spring и Hibernate.

Java широко применяется для создания серверной логики веб-приложений, где её возможности дополняются различными инструментами, включая Spring Framework.

**Основные аспекты Spring Boot для построения REST API**

Spring Boot — это популярный фреймворк для разработки приложений на Java, который упрощает создание REST API. Основные особенности Spring Boot:

* **Автоконфигурация**: минимизирует необходимость ручной настройки.
* **Встроенный веб-сервер**: поддержка Tomcat и Jetty позволяет запускать приложения без внешних серверов.
* **Поддержка REST**: встроенные аннотации, такие как @RestController и @RequestMapping, упрощают создание API.
* **Интеграция с базами данных**: Spring Data JPA обеспечивает удобную работу с реляционными базами данных.

Spring Boot ускоряет разработку, предоставляя готовые инструменты для создания REST API, таких как обработка запросов и форматирование ответов.

**Использование MySQL для хранения данных**

MySQL является одной из самых популярных реляционных баз данных благодаря своей надёжности и производительности. Преимущества использования MySQL:

* **Высокая производительность**: оптимизирован для обработки большого объёма данных.
* **Масштабируемость**: поддерживает масштабирование как вертикальное, так и горизонтальное.
* **Совместимость**: интегрируется с большинством современных технологий, включая Spring Boot.
* **Безопасность данных**: поддерживает механизм транзакций и репликации для защиты данных.

В проекте "Quiz Game" MySQL используется для хранения информации о пользователях, вопросах и результатах тестов. Для работы с базой данных применяется библиотека Spring Data JPA, которая упрощает реализацию CRUD-операций.

**1.2 Анализ существующих решений**

**Сравнение популярных приложений-викторин по языку Java**

Среди популярных приложений-викторин, ориентированных на обучение языку программирования Java, можно выделить следующие:

1. **SoloLearn**:
   * **Преимущества**: Предлагает интерактивные задания и тесты по Java, поддерживает геймификацию процесса обучения, доступно на разных устройствах.
   * **Недостатки**: Ограниченные возможности для создания собственных вопросов, отсутствие функций для запоминания материала через ассоциации.
2. **CodeGym**:
   * **Преимущества**: Углубленный курс Java с задачами и викторинами, поддержка уровней сложности.
   * **Недостатки**: Фокусируется больше на практических заданиях, чем на викторинах, что может быть сложным для новичков.
3. **Quizlet (наборы по Java)**:
   * **Преимущества**: Поддержка создания собственных наборов вопросов, возможность повторения материала через карточки.
   * **Недостатки**: Отсутствие специализированного контента по Java, неинтерактивный подход к обучению.

**Преимущества и недостатки аналогов**

Анализ приложений-викторин показывает, что они часто имеют как сильные стороны, так и ограничения. Например:

* **Преимущества**:
  + Возможность обучения через тесты и карточки (Quizlet).
  + Геймификация и интерактивность (SoloLearn).
  + Углублённый подход к обучению (CodeGym).
* **Недостатки**:
  + Отсутствие запоминающих элементов, таких как изображения-ассоциации.
  + Сложность интерфейса для новичков.

Проект "Quiz Game" нацелен на решение этих проблем за счёт простого и удобного интерфейса, который помогает новичкам в программировании тренировать свои теоретические знания в формате викторины.

**Глава 2. Практическая часть**

**2.1 Архитектура приложения**

**Описание модульной структуры приложения**

Приложение "Quiz Game" построено по модульному принципу, что упрощает поддержку и расширение функциональности. Основные модули:

1. **Модуль данных**:
   * Отвечает за взаимодействие с базой данных.
   * Использует Spring Data JPA для работы с сущностями, такими как вопросы и результаты тестов.
2. **Модуль логики (сервисы)**:
   * Реализует бизнес-логику приложения, включая проверку правильности ответов и расчёт результатов.
3. **Модуль API (контроллеры)**:
   * Обеспечивает взаимодействие клиента с сервером через REST API.
   * Включает эндпоинты для получения вопросов, отправки ответов и получения результатов.
4. **Клиентский модуль**:
   * Реализован с использованием HTML, CSS и JavaScript.
   * Отвечает за отображение интерфейса и взаимодействие с серверной частью через AJAX-запросы.

**Используемые технологии: Spring Boot, JPA, Docker**

1. **Spring Boot**:
   * Упрощает разработку серверной части приложения.
   * Предоставляет встроенные инструменты для реализации REST API и интеграции с базой данных.
2. **JPA (Java Persistence API)**:
   * Используется для взаимодействия с базой данных MySQL.
   * Упрощает реализацию операций с данными, таких как создание, чтение, обновление и удаление (CRUD).
3. **Docker**:
   * Используется для контейнеризации приложения.
   * Позволяет легко разворачивать приложение на разных платформах.
   * Упрощает настройку среды разработки и тестирования.

**2.2 Реализация функционала**

**Создание REST API для управления вопросами и пользователями**

Для реализации REST API в приложении "Quiz Game" используется Spring Boot. Основные эндпоинты API включают:

1. **Получение списка вопросов**:
   * Метод: GET /api/questions
   * Описание: Возвращает все доступные вопросы.
   * Пример ответа:
   * [
   * {
   * "id": 1,
   * "questionText": "Что такое Java?",
   * "answerA": "Язык программирования",
   * "answerB": "Операционная система",
   * "answerC": "База данных",
   * "answerD": "Редактор кода",
   * "correctAnswer": "A"
   * }
   * ]
2. **Отправка ответа пользователя**:
   * Метод: POST /api/submit
   * Описание: Принимает ответ пользователя на вопрос и возвращает результат.
   * Пример тела запроса:
   * {
   * "questionId": 1,
   * "selectedAnswer": "A"
   * }
   * Пример ответа:
   * {
   * "correct": true,
   * "correctAnswer": "A"
   * }

**Реализация обработки ответов и подсчёта результатов**

Серверная логика обработки ответов включает следующие шаги:

1. Получение вопроса из базы данных по идентификатору.
2. Сравнение ответа пользователя с правильным ответом.
3. Возврат результата с указанием правильности ответа.

Spring Boot и JPA используются для взаимодействия с базой данных, обеспечивая надёжную и производительную обработку данных.

**2.3 Интерфейс пользователя**

**Дизайн интерфейса**

Интерфейс приложения "Quiz Game" ориентирован на простоту и удобство использования. Основные элементы дизайна:

1. **Прогресс-бар**: отображает текущий прогресс пользователя в викторине.
2. **Область вопроса**: содержит текст вопроса и варианты ответов.
3. **Кнопки выбора ответа**: интуитивно понятны и реагируют на действия пользователя.
4. **Секция результатов**: отображает количество правильных ответов и общее число вопросов.

**Взаимодействие фронтенда с сервером**

Фронтенд приложения реализован с использованием JavaScript для обработки AJAX-запросов к серверу. Основные взаимодействия включают:

1. **Загрузка вопросов**: при запуске приложения фронтенд отправляет запрос на эндпоинт GET /api/questions и отображает полученные данные.
2. **Отправка ответа**: после выбора пользователем варианта ответа фронтенд отправляет запрос на эндпоинт POST /api/submit и обрабатывает результат.
3. **Обновление интерфейса**: в зависимости от ответа пользователя интерфейс обновляется, отображая корректность ответа и текущий прогресс.

**2.4 Пример использования**

**Примеры запросов к API**

1. **Запрос на получение всех вопросов**:
2. GET /api/questions HTTP/1.1
3. Host: localhost:8080
4. Accept: application/json

**Ответ**:

[

{

"id": 1,

"questionText": "Что такое Java?",

"answerA": "Язык программирования",

"answerB": "Операционная система",

"answerC": "База данных",

"answerD": "Редактор кода",

"correctAnswer": "A"

}

]

1. **Запрос на отправку ответа пользователя**:
2. POST /api/submit HTTP/1.1
3. Host: localhost:8080
4. Content-Type: application/json
5. {
6. "questionId": 1,
7. "selectedAnswer": "A"
8. }

**Ответ**:

{

"correct": true,

"correctAnswer": "A"

}

**Демонстрация интерфейса и функционала**

1. **Стартовый экран**: отображается название викторины и кнопка "Начать тестирование".
2. **Процесс тестирования**: вопросы и варианты ответов появляются последовательно, прогресс-бар обновляется.
3. **Результаты**: по завершении теста пользователю показывается количество правильных ответов и рекомендации по дальнейшему обучению.
4. **Пример взаимодействия**:
   * Пользователь нажимает кнопку ответа, данные отправляются на сервер.
   * Результат отображается мгновенно, прогресс обновляется.

Интерфейс интуитивно понятен, что делает приложение удобным даже для новичков.

**Заключение**

**Итоги**

* Разработано веб-приложение для тестирования знаний основ языка Java.
* Достигнуты поставленные цели: создано веб-приложение для тестирования знаний, которое успешно реализует заявленный функционал, включая REST API для обработки запросов и взаимодействие с базой данных. Решены задачи по проектированию архитектуры, разработке пользовательского интерфейса и тестированию системы, что подтверждает её готовность для дальнейшего использования и развития.

**Перспективы развития**

* Добавление новых типов вопросов.
* Замена статических картинок-ассоциаций на интерактивные анимации, которые реагируют на правильный или неправильный ответ пользователя, завершаясь позитивной или негативной концовкой для персонажа, что способствует вовлечению и улучшению процесса запоминания.
* Внедрение адаптивных алгоритмов тестирования.

## Список используемой литературы

### Литература

1. Блинов А. В., Романчик В. С. Java. Методы программирования. — СПб.: Питер, 2021. — 720 с.
2. Шилдт Г. Полный справочник по Java. — 11-е изд. — М.: Вильямс, 2020. — 960 с.
3. Эккель Б. Философия Java. — 4-е изд. — СПб.: Питер, 2020. — 1168 с.

### Онлайн-ресурсы

1. Документация по Spring Framework: https://spring.io/docs
2. Документация по MySQL: https://dev.mysql.com/doc/
3. Docker Documentation: https://docs.docker.com/
4. MDN Web Docs — справочник по HTML/CSS/JavaScript: https://developer.mozilla.org/
5. REST API Tutorial: https://restfulapi.net/

### Зависимости

1. Spring Boot (версии 2.6.6): https://spring.io/projects/spring-boot
2. Hibernate ORM (версии 5.6.7): https://hibernate.org/orm/
3. MySQL Connector/J (версии 8.0): https://dev.mysql.com/downloads/connector/j/
4. Docker Compose (версии 1.29): <https://docs.docker.com/compose/>
5. Maven Plugins: https://maven.apache.org/plugins/

### Дополнительные источники

1. Официальный сайт Quizlet: https://quizlet.com
2. Официальный сайт SoloLearn: https://www.sololearn.com
3. CodeGym: https://codegym.cc

# Приложения История команд 1 docker build -t quiz-app:1.0 .

# 2 docker run -p 8080:8080 quiz-app:1.0

# 3 docker run --name mysql -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=db\_quiz\_game -e MYSQL\_USER=user -e MYSQL\_PASSWORD=password -p 3306:3306 -d mysql:8.0

# 4 docker-compose down

# 5 docker-compose up --build -d

# 6 docker-compose down

# 7 docker ps

# 8 docker images

# 9 docker-compose up --build -d

# 10 docker-compose logs quiz-app

# 11 docker-compose down

# 12 docker-compose up --build -d

# 13 docker exec -it mysql bash

# 14 docker-compose down

# 15 docker-compose up --build -d

# 16 docker-compose down

# 17 docker-compose down

# 18 docker-compose up --build -d

# 19 docker-compose down

# 20 docker-compose up --build -d

# 21 docker exec -it mysql bash

# 22 docker-compose down

# 23 docker-compose up --build -d

# 24 docker-compose down

# 25 docker ps

# 26 docker images

# 27 docker-compose up --build -d

# 28 echo "# Diploma\_progect" >> README.md

# 29 git init

# 30 git add README.md

# 31 git commit -m "first commit"

# 32 git branch -M main

# 33 git remote add origin <https://github.com/KristiBu888/Diploma_progect.git>

# 34 git push -u origin main

