Geekbrains

**Разработка веб-приложения «Language Trainer» для тренировки английской орфографии на базе Java и Spring Framework**

Разработчик

Веб-разработка на Java

Шевяков Евгений Павлович

Нижний Новгород

2024

**Содержание**

[Введение 5](#_Toc181955123)

[Теоретические аспекты разработки приложения Language Trainer 7](#_Toc181955124)

[Понятие и функции тренажеров для изучения языков 7](#_Toc181955125)

[Особенности разработки приложений с использованием Java Spring 10](#_Toc181955126)

[Хранение данных и использование базы данных PostgreSQL 13](#_Toc181955127)

[Реализация MVC-архитектуры и использование шаблонизатора Thymeleaf 15](#_Toc181955128)

[Сортировка и работа со списками слов 17](#_Toc181955129)

[Сетевые особенности и настройка IP-адреса сервера 20](#_Toc181955130)

[Валидация и обработка ошибок 21](#_Toc181955131)

[Валидация данных: проверка на пустоту и корректность ввода 21](#_Toc181955132)

[Валидация через аннотации 22](#_Toc181955133)

[Обработка ошибок и исключений 22](#_Toc181955134)

[Обратная связь для пользователя 23](#_Toc181955135)

[Логирование и поддержка устойчивости 23](#_Toc181955136)

[Валидация и масштабируемость 23](#_Toc181955137)

[Применение аннотаций API для обработки ответов и ошибок 24](#_Toc181955138)

[Стандартизация ответов с помощью аннотаций API 24](#_Toc181955139)

[Обработка ошибок и кастомные аннотации для исключений 25](#_Toc181955140)

[Преимущества документации с помощью аннотаций API 25](#_Toc181955141)

[Улучшение читаемости и поддерживаемости кода 26](#_Toc181955142)

[Унификация обработки ответов и повышение надежности 26](#_Toc181955143)

[Снижение риска ошибок и упрощение тестирования 27](#_Toc181955144)

[Применение аннотаций API как стандарт разработки 27](#_Toc181955145)

[Заключение 28](#_Toc181955146)

[Практическая реализация приложения: архитектура, функциональность и интеграция технологий 30](#_Toc181955147)

[Подготовка среды и конфигурация базы данных 30](#_Toc181955148)

[Создание моделей данных 32](#_Toc181955149)

[Модель User 32](#_Toc181955150)

[Модель Word 33](#_Toc181955151)

[Преимущества аннотаций и ORM в модели данных 33](#_Toc181955152)

[Валидация и целостность данных 34](#_Toc181955153)

[Архитектура MVC и создание контроллеров 34](#_Toc181955154)

[Model: Определение моделей данных 34](#_Toc181955155)

[View: Представления на основе Thymeleaf 35](#_Toc181955156)

[Controller: Управление запросами и взаимодействие с пользователем 36](#_Toc181955157)

[Интеграция компонентов MVC 36](#_Toc181955158)

[Преимущества использования MVC 37](#_Toc181955159)

[Реализация шаблонов с использованием Thymeleaf 37](#_Toc181955160)

[Основные представления приложения 37](#_Toc181955161)

[Особенности использования Thymeleaf для внедрения данных 38](#_Toc181955162)

[Преимущества использования Thymeleaf для создания адаптивного интерфейса 39](#_Toc181955163)

[Реализация функций сортировки и использования паттерна «Стратегия» 40](#_Toc181955164)

[Применение паттерна «Стратегия» для сортировки данных 40](#_Toc181955165)

[Механизм выбора и применения стратегий сортировки 40](#_Toc181955166)

[Преимущества использования паттерна «Стратегия» для сортировки 41](#_Toc181955167)

[Аутентификация и авторизация с использованием Spring Security 42](#_Toc181955168)

[Валидация данных и обработка ошибок 44](#_Toc181955169)

[Документирование API с помощью аннотаций OpenAPI 46](#_Toc181955170)

[Заключение 47](#_Toc181955171)

[Заключение 49](#_Toc181955172)

[Оценка проведённого исследования и его результатов 49](#_Toc181955173)

[Практическая значимость и рекомендации 50](#_Toc181955174)

[Общий итог и планы на дальнейшее совершенствование 50](#_Toc181955175)

[Список используемой литературы 51](#_Toc181955176)

# Введение

Современные технологии играют ключевую роль в обучении и развитии навыков. Изучение иностранных языков, включая английский, является одной из важнейших задач в образовательных учреждениях и в жизни каждого человека. В условиях глобализации и активного обмена информацией владение иностранными языками становится необходимым для успешной профессиональной и личной жизни. Одним из самых эффективных способов изучения языка является регулярное повторение слов и фраз, что способствует их запоминанию и улучшению навыков письма.

Технологии, такие как мобильные приложения, веб-сервисы и интерактивные платформы, позволяют сделать процесс обучения более доступным, интересным и эффективным. Существующие решения на основе мобильных приложений и веб-сайтов помогают обучающимся быстро осваивать новые слова и фразы с использованием различных методов и техник. Одним из популярных методов является тренировка написания слов, что способствует улучшению как письменной речи, так и общей языковой грамотности.

В рамках данной работы будет разработано веб-приложение, которое будет использоваться для тренировки в написании английских слов. Приложение будет предоставлять пользователю русское слово, и он должен будет ввести его английский перевод. Также будет реализована возможность добавления собственных слов и их перевода. Разработка осуществляется с использованием фреймворка Java Spring, который предоставляет мощные инструменты для создания веб-приложений, обеспечивая надежность, безопасность и масштабируемость.

Целью данной дипломной работы является создание веб-приложения для тренировки написания английских слов. В ходе работы будет спроектирована и реализована архитектура приложения, выбран набор технологий и инструментов, а также обеспечена интеграция с базой данных для хранения информации о словах и пользователях. Работа включает в себя также реализацию механизма аутентификации.

Задачами работы являются:

* Разработка архитектуры backend-части приложения с использованием фреймворка Java Spring.
* Реализация функционала для работы с базой данных, включая добавление, редактирование, удаление и сортировку слов.
* Разработка механизма аутентификации пользователей с использованием Spring Security.
* Создание фронтенд-части с использованием шаблонизатора Thimleaf.
* Тестирование и отладка приложения.

Важной частью работы является также создание интерфейса для взаимодействия с пользователем, что обеспечит удобство использования системы. В ходе разработки будут рассмотрены вопросы безопасности, производительности и масштабируемости системы.

# Теоретические аспекты разработки приложения Language Trainer

## Понятие и функции тренажеров для изучения языков

Современные языковые тренажёры являются мощным и востребованным инструментом для изучения иностранных языков. Благодаря стремительному развитию технологий, такие тренажёры представляют собой интерактивные приложения, которые охватывают практически все аспекты языкового обучения – от чтения и письма до аудирования, произношения и увеличения словарного запаса. Важно отметить, что разнообразие подходов, предлагаемых этими программами, позволяет обучающимся подобрать наиболее подходящий метод для их целей и уровня знаний. Таким образом, языковые тренажёры активно применяются не только в самостоятельном обучении, но и в образовательных учреждениях, помогая преподавателям разнообразить учебный процесс и сделать его более гибким и увлекательным.

Программные языковые тренажеры отличаются друг от друга по своим функциональным возможностям и направленности. Некоторые из них ориентированы на обучение грамматике и синтаксису, другие помогают развивать навыки аудирования, а третьи направлены на улучшение произношения. Однако особое внимание стоит уделить тренажёрам, разработанным для совершенствования навыков письма и правильного написания слов, таких как приложение Language Trainer. Такие программы особенно важны для пользователей, стремящихся к высокому уровню языковой компетенции, поскольку они фокусируются на точной орфографии и запоминании значительного объема лексики. Правильное написание слов – один из основополагающих компонентов уверенного использования языка, особенно в академических и профессиональных контекстах.

Language Trainer – это приложение, ориентированное на русскоязычную аудиторию, целью которого является развитие навыков написания английских слов. Данное приложение предлагает пользователю ввести английский перевод для указанного русского слова, проверяя точность и корректность перевода. Такая методика полезна не только для расширения словарного запаса, но и для улучшения памяти, поскольку регулярное повторение английских слов с их переводом помогает создавать прочные ассоциативные связи между словами родного и иностранного языка. Это важно, поскольку позволяет пользователю быстрее и точнее вспоминать нужные слова, что необходимо для уверенного общения и написания на английском языке.

Современные исследования в области когнитивной психологии показывают, что регулярное повторение является ключевым фактором при запоминании информации. В частности, в языковом обучении регулярное повторение помогает переводить информацию из кратковременной памяти в долговременную, благодаря чему знания становятся более устойчивыми и легко восстанавливаемыми. Приложения, подобные Language Trainer, создают оптимальные условия для регулярного закрепления лексики, что положительно влияет на когнитивные процессы и улучшает общее восприятие языка. Постоянная практика способствует также снижению уровня тревожности при использовании иностранного языка, повышая уверенность в себе.

Language Trainer отличается еще и тем, что пользователи могут добавлять в приложение свои собственные слова для тренировок. Это позволяет им адаптировать процесс обучения под свои конкретные потребности и задачи, например, тренировать лексику, актуальную для работы или учёбы. Добавление собственных слов способствует личностно-ориентированному подходу к изучению языка, что делает обучение более значимым и эффективным. В результате пользователи могут создать свой персональный словарь, который отражает их интересы и профессиональные задачи, что значительно улучшает мотивацию к обучению.

Кроме того, такие тренажёры, как Language Trainer, предоставляют различные формы обратной связи, благодаря чему пользователи сразу узнают о допущенных ошибках и могут работать над их исправлением. Это позволяет не только оперативно исправлять ошибки, но и более детально анализировать свои слабые стороны. Например, если пользователь часто ошибается в написанииопределённых слов, он может сфокусироваться именно на этой категории, что делает процесс обучения более целенаправленным и продуктивным. Интерактивный подход позволяет каждому пользователю работать в собственном темпе и уделять больше внимания тем аспектам, которые требуют дополнительного закрепления.

Приложения, ориентированные на развитие навыков письма, играют важную роль в языковом обучении, поскольку навыки письма и орфографии являются фундаментом для уверенного владения языком. Письменные навыки особенно важны в контексте академического обучения и профессиональной деятельности, где точность и корректность написания имеют значительное значение. Language Trainer позволяет пользователям не только расширить лексический запас, но и развить навык правильного и быстрого написания слов, что облегчает процесс запоминания сложных лексических конструкций.

Приложение имеет важную функцию, которая позволяет пользователям повторять слова, не использованные в тренировках длительное время. Эта функция ориентирована на поддержку долговременной памяти и помогает избежать забывания выученной лексики. Приложение анализирует историю повторений каждого слова и при обнаружении, что определённые слова не повторялись на протяжении более чем нескольких дней, включает их в список «забытых» слов, предлагая пользователю вернуться к их тренировке. Это особенно важно для пользователей, которые стремятся не только накапливать новый словарный запас, но и удерживать уже изученные слова на уровне уверенного использования. Регулярное повторение таких слов способствует их более прочному закреплению в памяти и увеличивает шанс, что они будут легко восстановлены при необходимости.

Эта функция позволяет пользователям ориентироваться на актуальные потребности в изучении языка и закрывает одну из ключевых проблем языковой практики – постепенное забывание материала, который не использовался. Возможность вернуться к давно не повторяемым словам помогает создать более устойчивую и организованную систему для изучения и повторения слов, делая процесс обучения более последовательным и структурированным. Более того, функция «забытых» слов помогает пользователям сохранять мотивацию, поскольку регулярное возвращение к материалу, ранее уже освоенному, повышает уверенность в знаниях и демонстрирует, насколько успешно идёт процесс изучения.

Таким образом, приложение Language Trainer предлагает многоуровневый подход к обучению, объединяющий в себе элементы самостоятельного обучения, повторения и контроля. Это не просто программа для заучивания слов, а полноценный языковой тренажёр, который помогает пользователю уверенно и последовательно совершенствовать свои навыки владения английским языком. Регулярное использование таких приложений позволяет значительно ускорить процесс запоминания, облегчить процесс адаптации к иностранному языку и укрепить уверенность в своих силах.

Современные языковые тренажеры стали неотъемлемой частью процесса обучения, обеспечивая пользователям поддержку на всех уровнях, от начального до продвинутого. Они позволяют расширять словарный запас, совершенствовать грамматику и улучшать навыки аудирования и произношения. Приложения, которые направлены на тренировку навыков письма, играют важную роль в подготовке пользователей к уверенной работе с языком, что особенно актуально для тех, кто стремится к свободному владению иностранным языком на высоком уровне.

## Особенности разработки приложений с использованием Java Spring

**Spring Framework** является одним из самых востребованных фреймворков для создания приложений на Java, благодаря своей гибкости, масштабируемости и богатому набору возможностей. Он предоставляет разработчикам обширный функционал для построения современных веб-приложений, поддерживая такие важные концепции, как инверсия управления (IoC), аспектно-ориентированное программирование (AOP) и возможность интеграции с разными источниками данных. **Spring** также реализует модульную архитектуру, что позволяет разработчикам использовать только те компоненты, которые действительно необходимы для проекта, будь то работа с веб-запросами, обработка транзакций, безопасность или интеграция с другими библиотеками и сервисами. Эта модульность делает Spring Framework подходящим для создания как небольших приложений, так и крупных корпоративных систем.

Для разработки **Language Trainer** былвыбран **Spring Boot**, который является надстройкой над Spring Framework и значительно упрощает процесс конфигурации и развертывания приложений. Spring Boot автоматически настраивает базовые компоненты, позволяя разработчикам минимизировать количество ручных настроек и сосредоточиться на бизнес-логике приложения. Важной особенностью Spring Boot является его способность поддерживать встроенные серверы, такие как Tomcat и Jetty, что позволяет быстро запускать приложение локально и тестировать его работу без необходимости в сложной серверной инфраструктуре. В случае **Language Trainer** это стало особенно полезным, так как позволило ускорить процесс разработки и тестирования, сделав цикл внедрения изменений более быстрым и эффективным.

Для работы с базой данных в **Language Trainer** был использован **Spring Data JPA,** который предоставляет удобный и гибкий способ взаимодействия с базами данных через объектно-реляционное отображение (ORM). Этот модуль поддерживает построение SQL-запросов и их автоматическую генерацию, что позволяет разработчикам работать с данными на уровне Java-объектов, не прибегая к написанию сложного SQL-кода. Это особенно полезно для создания удобных, легко поддерживаемых и тестируемых репозиториев данных. В Language Trainer, Spring Data JPA играет ключевую роль, обеспечивая эффективное управление словарной базой, в которой хранится информация о пользователях и списке изучаемых слов. ORM-стратегии позволяют разрабатывать функционал для хранения и редактирования данных таким образом, чтобы минимизировать риск ошибок при взаимодействии с базой данных.

Важным элементом в реализации пользовательского интерфейса **Language Trainer** является использование шаблонизатора **Thymeleaf**. В отличие от традиционных JSP, Thymeleaf предлагает гибкие возможности для разработки адаптивных и динамических HTML-страниц, что позволяет легко интегрировать данные, полученные от сервера, с пользовательским интерфейсом. Использование Thymeleaf делает процесс разработки UI более удобным и интуитивно понятным, так как он поддерживает привязку данных напрямую в HTML, упрощая создание форм для взаимодействия пользователя с приложением. Это особенно важно для Language Trainer, так как пользователи регулярно вводят ответы на cлова и работают с персонализированным контентом. Адаптивность интерфейса на базе Thymeleaf также позволяет сделать приложение более удобным для работы на разных устройствах и экранах.

Для обеспечения безопасности пользовательских данных в приложении используется **Spring Security** – модуль Spring, который поддерживает гибкую настройку аутентификации и авторизации. В **Language Trainer** этот модуль реализует защиту учётных записей пользователей, ограничивая доступ к функционалу приложения только для авторизованных пользователей. Spring Security предлагает богатый набор инструментов для защиты данных и конфиденциальности, включая поддержку шифрования паролей, работу с различными протоколами авторизации, интеграцию с внешними провайдерами аутентификации и настройку ролевой модели. Применение Spring Security позволяет обеспечить защиту личных данных, делая процесс использования приложения безопасным и конфиденциальным для всех пользователей.

Сочетание всех этих компонентов делает **Language Trainer** надёжным, функциональным и удобным в использовании приложением для пользователей. Благодаря модульной структуре Spring Framework, автоматизации Spring Boot, гибкому взаимодействию с базой данных через Spring Data JPA, удобному UI на базе Thymeleaf и безопасности Spring Security, приложение обладает гибкой и надежной архитектурой.

## Хранение данных и использование базы данных PostgreSQL

Для хранения данных в приложении **Language Trainer** выбрана реляционная база данных **PostgreSQL**. Этот выбор обусловлен высокой надёжностью, масштабируемостью и мощными возможностями PostgreSQL, которая, как одна из ведущих систем управления базами данных (СУБД), предоставляет разработчикам широкий спектр функций для работы с данными. Одним из главных преимуществ PostgreSQL является её поддержка ACID-транзакций, что обеспечивает целостность данных даже при параллельных операциях записи и чтения. Это особенно важно для приложений, где необходимо гарантировать согласованность данных пользователей и их словарей, а также защиту от возможных сбоев. Кроме того, PostgreSQL поддерживает индексы, что значительно ускоряет операции поиска, а также позволяет оптимизировать обработку больших объемов данных, что делает её подходящей для приложений с высоким уровнем запросов и больших объёмов информации.

В приложении **Language Trainer** данные о пользователях и словах сохраняются с помощью **Spring Data JPA –** технологии, которая позволяет взаимодействовать с базой данных через объектно-реляционное отображение (ORM), используя объектно-ориентированный подход. Это означает, что разработчик может работать с данными на уровне Java-объектов, а Spring Data JPA автоматически преобразует эти объекты в SQL-запросы и управляет взаимодействием с базой данных. Использование ORM также позволяет минимизировать количество ручного SQL-кода и упростить процесс работы с данными. В данном случае объекты, представляющие пользователей и слова, хранятся в таблицах базы данных, и для их управления используется репозиторий, реализованный через JPA.

Для автоматизации работы с базой данных в **Language Trainer** используется настройка hibernate.ddl-auto с параметром update, что позволяет Hibernate автоматически обновлять структуру базы данных при изменении моделей данных в коде. Это означает, что, когда в приложение добавляются новые поля или изменяются существующие, Hibernate автоматически синхронизирует эти изменения с базой данных, избегая необходимости вручную изменять SQL-схему. Данная опция позволяет разработчику быстро и безболезненно вносить изменения в структуру данных, что особенно полезно на этапе разработки и тестирования, когда структура моделей данных может часто изменяться.

Для описания моделей данных в **Language Trainer** применяются аннотации, такие как @Entity, @Table, @Id и @Column, которые позволяют точно описывать структуру и отношения между таблицами базы данных. Например, сущности User и Word определяют таблицы в базе данных, где хранится информация о пользователях и их словах. Аннотация @Entity используется для обозначения Java-класса как сущности базы данных, @Table указывает название таблицы, @Id определяет первичный ключ, а @Column позволяет задавать параметры полей таблицы. Благодаря этому подходу, процесс создания и управления данными становится интуитивно понятным и хорошо структурированным, что облегчает дальнейшую поддержку и масштабирование приложения.

Кроме того, PostgreSQL поддерживает работу с различными типами данных, что позволяет разработчикам гибко управлять структурой таблиц и делать её максимально адаптивной к различным задачам. Например, для хранения данных о пользователях можно использовать типы данных TEXT или VARCHAR, для слов – VARCHAR или TEXT, а для других числовых значений – INTEGER или BIGINT. Поддержка различных типов данных делает PostgreSQL удобной для реализации сложных структур, таких как хранение дат, значений булевых типов или уникальных идентификаторов, которые могут понадобиться при расширении функционала приложения.

Кроме того, PostgreSQL предоставляет гибкие инструменты для управления правами доступа, что позволяет ограничить доступ к определённым данным для разных ролей пользователей. В Language Trainer это может быть использовано для разделения прав доступа администратора и обычного пользователя, чтобы обеспечить дополнительную защиту данных.

## Реализация MVC-архитектуры и использование шаблонизатора Thymeleaf

Language Trainer построен на основе архитектуры MVC (Model-View-Controller), что обеспечивает гибкое, поддерживаемое и хорошо структурированное приложение. Подход MVC разделяет ключевые аспекты приложения на три отдельных компонента: Model (модель), View (представление) и Controller (контроллер). Такое разделение позволяет улучшить организацию кода и облегчить его дальнейшее развитие и поддержку, а также упрощает тестирование и внесение изменений в код, так как каждый компонент отвечает за конкретный аспект функционирования приложения.

Model (Модель) – это слой, представляющий данные приложения, которые включают информацию о пользователях и их словарях. В контексте Language Trainer, модель управляет такими объектами, как User и Word, которые описывают пользователя и его слова, а также их взаимосвязи. Например, модель Word хранит информацию о словах, их переводах и времени последнего использования, что позволяет отслеживать прогресс пользователя. Благодаря аннотациям JPA, Model автоматически маппируется в таблицы базы данных PostgreSQL, обеспечивая лёгкую интеграцию с СУБД и упрощая хранение данных.

View (Представление) – это слой, отвечающий за отображение данных и за взаимодействие с пользователем. В Language Trainer для создания представлений используется Thymeleaf, который тесно интегрируется с Spring Framework. Thymeleaf позволяет динамически внедрять данные прямо в HTML-страницы, что упрощает процесс создания адаптивного пользовательского интерфейса. Представления включают такие страницы, как страница с текущими словами для изучения, страницы входа и регистрации, а также интерфейсы для добавления новых слов или управления настройками. Например, при отображении списка слов, которые пользователь давно не повторял, Thymeleaf автоматически вставляет в HTML соответствующие данные, такие как сами слова, их переводы и другую информацию, предоставленную моделью. Такая гибкость позволяет быстро обновлять и персонализировать интерфейс, делая взаимодействие с приложением интуитивным и удобным.

Controller (Контроллер) – это компонент, обеспечивающий связь между пользователем и системой. В Language Trainer контроллеры отвечают за обработку запросов пользователя, получение данных от модели и передачу их в представление. Контроллеры включают методы для отображения слов, добавления новых слов, обновления информации о пользователях, а также другие действия, такие как вход и регистрация. Например, когда пользователь отправляет запрос на добавление нового слова, контроллер получает данные из формы, проверяет их, и, если они корректны, передаёт в модель для сохранения в базе данных. В случае успешной обработки контроллер направляет пользователя на страницу со списком слов, что создает плавный и последовательный пользовательский опыт.

Использование Thymeleaf в качестве шаблонизатора для представлений предоставляет Language Trainer возможность создания страниц с динамическими элементами. Thymeleaf поддерживает специальные атрибуты для работы с переменными, циклами и условиями, что позволяет разработчику гибко управлять контентом. Например, для списка слов можно создать страницу, где каждый элемент списка динамически формируется на основе данных, полученных от контроллера. Thymeleaf позволяет не только отображать данные, но и выполнять проверки и валидации, например, выводя сообщения об ошибках при неправильном вводе данных в форму.

Благодаря MVC-архитектуре Language Trainer приобретает структуру, которая упрощает масштабирование и добавление новых функций. Например, если требуется добавить новый функционал, такой как уведомления о том, что слово давно не повторялось, это можно сделать, расширив модель и контроллер, которые будут обрабатывать логику напоминаний, а также добавив новые элементы в представление. Сложные взаимодействия между пользователем и системой, такие как выполнение поиска или фильтрации, также легко реализуются через добавление соответствующих методов в контроллере и использование Thymeleaf для отображения результатов на странице.

Вместе все три компонента MVC делают приложение гибким, удобным и поддерживаемым. Model упрощает работу с базой данных и обеспечивает безопасность данных, View обеспечивает удобный интерфейс и взаимодействие, а Controller позволяет обрабатывать запросы и управлять данными.

## Сортировка и работа со списками слов

Language Trainer предоставляет пользователям гибкие возможности для работы с их словарём, в частности, с функцией сортировки, которая улучшает удобство пользования приложением. Сортировка – важная составляющая для всех пользователей, особенно для тех, кто активно работает со словами, так как она позволяет структурировать и быстро находить нужную информацию. В Language Trainer реализованы функции сортировки по английским и русским словам, что значительно упрощает доступ к необходимым данным и повышает продуктивность при изучении языка.

Реализация сортировки основана на паттерне проектирования «Стратегия», который предполагает создание отдельных классов для каждой стратегии сортировки, чтобы можно было легко изменять поведение программы без необходимости модифицировать основной код. В данном случае, в Language Trainer для сортировки по английским и русским словам реализованы две стратегии: EnglishSortStrategy и RussianSortStrategy. Каждый из этих классов предоставляет собственный метод сортировки, что позволяет пользователю выбрать нужный вариант отображения словаря – по английским словам (в алфавитном порядке) или по русским переводам. Это даёт пользователю возможность гибко настраивать способ отображения данных под свои нужды, будь то повторение слов по алфавиту или использование русского перевода в качестве отправной точки.

Схема работы с сортировкой в Language Trainer выглядит следующим образом:

Выбор стратегии сортировки. При взаимодействии с приложением пользователь имеет возможность выбрать, по какому критерию должен быть отсортирован список слов – по английским словам или по русским переводам. Это действие инициирует выбор соответствующей стратегии сортировки. Если пользователь выбирает сортировку по английским словам, приложение использует EnglishSortStrategy. Если же пользователь предпочитает сортировку по русским словам, включается RussianSortStrategy.

Внедрение стратегии. После того как пользователь выбрал нужную стратегию, приложение, основываясь на паттерне «Стратегия», применяет выбранный алгоритм сортировки к текущему списку слов. Это достигается тем, что для каждого случая заранее определены соответствующие классы (стратегии), которые имеют одинаковый интерфейс и реализуют метод сортировки. Таким образом, код, отвечающий за сортировку, становится гибким и легко поддерживаемым, так как изменение логики сортировки или добавление новых методов не требуют изменения основного кода. Например, если в будущем будет необходимо добавить сортировку по дате добавления слова или по частоте его использования, достаточно создать новый класс, реализующий соответствующий алгоритм сортировки, и интегрировать его в систему.

Применение сортировки и отображение результатов. После выбора стратегии данные в списке слов сортируются в соответствии с выбранными параметрами, а результат отображается пользователю в удобной форме на экране. Используя Thymeleaf как шаблонизатор, Language Trainer динамически обновляет страницу, чтобы отобразить результаты сортировки. Это позволяет пользователям сразу видеть результат своего выбора и при необходимости быстро переключаться между разными видами сортировки, чтобы получить нужные данные.

Преимущества подхода с использованием паттерна «Стратегия» в Language Trainer:

Гибкость и масштабируемость. Паттерн «Стратегия» делает код более гибким и масштабируемым. При необходимости разработчики могут легко добавить новые методы сортировки, не меняя основную логику работы приложения.

Простота поддержки. Каждый алгоритм сортировки инкапсулирован в отдельный класс, что упрощает тестирование и отладку. В случае необходимости изменения одного алгоритма сортировки, изменения могут быть внесены только в соответствующий класс стратегии, что исключает влияние на другие части системы.

Переиспользование кода. Стратегии можно многократно использовать в разных частях приложения или в других проектах, где может понадобиться аналогичная функциональность.

Дополнительные функции сортировки и фильтрации, реализованные с использованием паттерна «Стратегия», позволяют пользователям не только структурировать данные по их предпочтениям, но и обеспечивают удобство работы с большими объёмами информации. В будущем, возможно добавление новых стратегий сортировки, таких как сортировка по дате добавления слова или сортировка по частоте ошибок, допущенных пользователем. Это ещё больше расширит возможности приложения, предоставляя пользователю полный контроль над структурированием своего словаря.

Использование подхода с выбором стратегии также способствует лучшему пользовательскому опыту, так как каждый пользователь может настроить приложение под свои уникальные потребности и привычки в изучении слов. Такой подход особенно полезен для продвинутых пользователей, которым требуется удобный способ отслеживания прогресса и регулярного повторения слов. Language Trainer, благодаря гибкой и модульной системе сортировки, дат возможность поддерживать структуру данных и расширять функциональность без изменения основного кода, что делает приложение мощным и адаптируемым к запросам пользователей.

## Сетевые особенности и настройка IP-адреса сервера

В процессе разработки приложения, важным аспектом является настройка сетевых параметров, особенно в контексте подключения к серверу. Для корректной работы приложения в многопользовательской среде требуется точная настройка сетевых интерфейсов и правильное определение IP-адреса сервера. В этом приложении используется интерфейс DataServer, который автоматически определяет IP-адрес сервера, на котором развёрнуто приложение. Эта информация позволяет не только правильно настроить взаимодействие пользователей с сервером, но и упрощает диагностику и поддержку инфраструктуры приложения.

Использование DataServer помогает обеспечивать связь между клиентами и сервером, предоставляя необходимую информацию о сети, что особенно важно при обслуживании и тестировании приложения. С помощью данного интерфейса можно определить локальный и внешний IP-адрес сервера, что необходимо для настройки внешних подключений и обеспечения безопасности сети. Например, при работе с внешними API или при конфигурации брандмауэра, важно точно указать IP-адрес, чтобы правильно настроить доступ к сервисам.

Кроме того, использование DataServer дает возможность мониторить состояние сети в реальном времени, что способствует более быстрому выявлению и устранению сетевых проблем. В ходе эксплуатации приложения можно будет оперативно получать информацию о возможных сбоях в подключении, изменениях в IP-адресах или других сетевых параметрах, что важным образом влияет на стабильность работы сервера. Это также полезно для тестирования приложений в различных сетевых условиях, так как позволяет моделировать различные сценарии и удостовериться в корректности работы приложения в условиях реальной сети.

Наконец, настройка IP-адреса сервера играет ключевую роль в организации безопасного и стабильного взаимодействия между сервером и клиентами. При наличии нескольких серверов в сети или при работе через виртуальные машины необходимо чётко настроить маршрутизацию и порты для каждого из них. Это позволяет избежать конфликтов и обеспечить правильное распределение трафика, особенно в случаях, когда приложение используется в многопользовательском режиме с высокой нагрузкой.

## Валидация и обработка ошибок

Валидация входных данных и обработка ошибок – важные компоненты для обеспечения надёжной и безопасной работы приложения, предотвращающие возможные ошибки пользователя и повышающие удобство взаимодействия с системой. В данном приложении валидируются данные, вводимые пользователем, такие как английский перевод слова, чтобы избежать распространённых ошибок, таких как ввод пустой строки или случайные пробелы. Это предотвращает ситуации, когда пользователь может случайно нажать клавишу Enter, добавив пустое слово в свой словарь, что могло бы создать неудобства при дальнейшем использовании приложения.

### Валидация данных: проверка на пустоту и корректность ввода

На уровне валидации приложение проверяет каждое введённое слово на наличие данных, чтобы избежать обработки пустых или некорректных значений. Это достигается путём простой, но эффективной проверки, которая отсеивает строки, содержащие только пробелы или не содержащие символов вовсе. Такой подход особенно полезен при добавлении новых слов в словарь, поскольку гарантирует, что каждое сохраненное слово действительно имеет смысловую нагрузку и будет полезно пользователю.

Если пользователь пытается добавить слово или его перевод, оставив одно из полей пустым, приложение направляет его обратно на форму ввода, не учитывая попытку как успешную или неудачную, и, при этом, отображает соответствующее уведомление о необходимости корректного заполнения всех полей. Это позволяет пользователю легко исправить ошибку и вводить данные повторно, не прерывая процесс обучения.

### Валидация через аннотации

Для более глубокой валидации, в Language Trainer используются аннотации, предоставляемые Spring Framework. Например, аннотация @NotEmpty применяется к полям, требующим обязательного заполнения, например, полям для ввода английского и русского слов. Таким образом, на этапе ввода данных проверяется, что значения действительно содержат текст, а не просто пробелы или пустую строку. При некорректном вводе система возвращает сообщение об ошибке, поясняя, что поле не может быть пустым, что улучшает пользовательский опыт и упрощает исправление ошибок.

### Обработка ошибок и исключений

Для корректной обработки возможных ошибок и исключительных ситуаций в приложении реализована система управления исключениями. Например, если пользователь пытается получить доступ к слову, которое отсутствует в базе данных (возможно, оно было удалено или недоступно по другой причине), приложение бросает исключение NotFoundWordException. Этот класс исключения обрабатывает ситуации, когда нужное слово не найдено, и предоставляет пользователю понятное сообщение об ошибке, что позволяет избежать неожиданных сбоев и повышает удобство использования.

В дополнение к NotFoundWordException, могут быть добавлены и другие пользовательские исключения для обработки конкретных ситуаций, таких как ошибки при обращении к базе данных, нарушение целостности данных, неверные идентификаторы и другие исключительные случаи. Обработка таких ошибок в одном месте позволяет централизовать контроль над ошибками и делать приложение более предсказуемым и устойчивым к непредвиденным ситуациям.

### Обратная связь для пользователя

Одним из приоритетов разработки было сделать Language Trainer максимально понятным и удобным для пользователя, включая обработку ошибок. При возникновении ошибок приложение отображает сообщения, помогающие пользователю понять причину проблемы и необходимые действия для её исправления. Например, если пользователь вводит пустую строку при добавлении перевода, на экране выводится уведомление о необходимости ввода данных. Если же запрашиваемое слово не найдено, пользователь увидит сообщение с предложением проверить правильность ввода.

Такая обратная связь способствует лучшему пользовательскому опыту, так как помогает сразу понять и исправить ошибку, не выходя из интерфейса приложения. Кроме того, предупреждения об ошибках настроены так, чтобы минимизировать неприятные ситуации, в которых пользователь может потерять данные из-за ошибки в работе программы.

### Логирование и поддержка устойчивости

Для повышения надежности приложения и упрощения отладки в случае возникновения ошибок ведётся логирование всех критических операций и ошибок. При каждой ошибке система записывает информацию о её природе, что позволяет разработчикам быстро обнаружить проблему и внести необходимые изменения в код. Это особенно полезно при обнаружении редких или специфичных ошибок, которые сложно воспроизвести в тестовой среде.

Например, если возникает NotFoundWordException, информация о запросе, вызвавшем ошибку, и контексте, в котором она возникла, будет записана в журнал. Это помогает не только понять природу проблемы, но и выявить слабые места в системе, где могут требоваться дополнительные проверки или улучшения.

### Валидация и масштабируемость

Подход к валидации и обработке ошибок, реализованный в приложении, легко масштабируется и может быть расширен по мере добавления новых функций и возможностей. Например, в будущем может быть добавлена более строгая валидация на этапе добавления слов, с проверкой орфографии или формата ввода. Также может быть реализована поддержка более сложных исключений, таких как ошибка соединения с базой данных, чтобы приложение могло корректно функционировать даже в случае временных технических проблем.

Таким образом, валидация и обработка ошибок не только помогают избежать простых ошибок, таких как ввод пустых значений, но и предоставляют механизм для реагирования на сложные и непредвиденные ситуации. Эти компоненты создают устойчивую и удобную среду, в которой пользователи могут сосредоточиться на изучении английского языка, не отвлекаясь на технические сложности.

## Применение аннотаций API для обработки ответов и ошибок

Аннотации API – это мощный инструмент для стандартизации ответов и обработки ошибок, особенно при работе с RESTful сервисами. В приложении они играют важную роль в обеспечении надежности и предсказуемости API, а также упрощают разработку, делая код более понятным и поддерживаемым. Кастомные аннотации, такие как OkList, OkWord, NotFoundResponse, и InternalServerError, помогают не только унифицировать ответы, но и детально документировать их, что важно для комплексных приложений, где ошибки и различные статусы ответа могут быть многочисленными.

### Стандартизация ответов с помощью аннотаций API

Аннотации, такие как OkList и OkWord, задают стандартные ответы API для успешных запросов, что упрощает понимание кода и поддержание единообразного стиля в приложении. Аннотация OkList используется для обозначения успешного возвращения списка данных, например, когда пользователь запрашивает все слова, сохраненные в своем словаре. Эта аннотация сообщает разработчикам и пользователям API, что при успешном запросе сервер вернёт список данных со статусом 200 OK. Использование OkWord аналогично, но применяется для успешного возвращения конкретного слова, когда запрашивается один элемент из базы данных. Такой подход снижает вероятность ошибок и упрощает поддержку: разработчики видят заранее обозначенные стандартные ответы и знают, какие данные должны быть возвращены для каждого сценария.

### Обработка ошибок и кастомные аннотации для исключений

При работе с API критически важно не только обрабатывать ошибки, но и ясно их документировать. Аннотация NotFoundResponse используется для указания на случай, когда запрашиваемый ресурс (например, слово) не найден. Этот сценарий обрабатывается с возвращением кода ошибки 404 Not Found и пояснением для пользователя, почему возникла ошибка. Такая аннотация помогает при разработке, так как позволяет централизованно управлять сценариями с отсутствием данных, обеспечивая более информативные и корректные ответы. При получении ошибки 404 пользователь может увидеть сообщение, поясняющее, что искомый ресурс не существует, что снижает путаницу и улучшает пользовательский опыт.

Аннотация InternalServerError отвечает за обработку непредвиденных серверных ошибок. Если в ходе выполнения запроса возникает ошибка, такая как сбой базы данных или исключение в бизнес-логике, эта аннотация позволяет API возвращать код ошибки 500 Internal Server Error, информируя, что возникла внутренняя ошибка на стороне сервера. Кастомные сообщения и обработчики, связанные с этой аннотацией, помогают точно определить причину ошибки и позволяют избежать слишком общих или вводящих в заблуждение сообщений для пользователей.

### Преимущества документации с помощью аннотаций API

Использование кастомных аннотаций также способствует улучшению автоматической документации API в рамках OpenAPI. В приложении, где аннотации OkList, OkWord, NotFoundResponse, и InternalServerError описывают возможные ответы, это облегчает создание понятной и точной спецификации API. Такой подход позволяет разработчикам автоматически генерировать документацию, которая ясно показывает, какие ответы могут возникнуть при различных запросах. Это значительно упрощает процесс интеграции и тестирования, так как документация не требует ручной правки и всегда актуальна, в отличие от статических описаний. Инструменты, поддерживающие OpenAPI, автоматически распознают аннотации, добавляют их в документацию и предоставляют разработчикам и пользователям API точные сведения о возможных ответах, их типах и структурах.

### Улучшение читаемости и поддерживаемости кода

Читаемость кода – одна из важных целей при использовании аннотаций. Когда каждый метод сопровождается стандартной аннотацией, такой как OkList или NotFoundResponse, становится легко понять, какой тип ответа следует ожидать от данного метода и как он будет обработан. Это особенно важно при командной разработке, так как каждый разработчик может быстро ориентироваться в коде, не изучая детально его реализацию. Аннотации помогают стандартизировать структуру ответов и обрабатываемые ошибки, что, в свою очередь, упрощает обучение новых членов команды и делает код более устойчивым к изменениям. Например, при необходимости изменить ответ API, достаточно обновить аннотацию, что предотвращает распространение ошибки по всему коду и сохраняет высокую степень согласованности.

### Унификация обработки ответов и повышение надежности

Еще одним важным аспектом применения аннотаций является унификация обработки ответов. В приложении это помогает поддерживать единый подход к возврату данных пользователям. Все методы контроллеров, например, возвращают данные или ошибки в стандартизированном формате, что предотвращает потенциальные сбои и недоразумения. Такая унификация гарантирует, что каждый ответ будет соответствовать ожидаемому шаблону и не вызовет проблем в процессе взаимодействия между компонентами системы. API становится более предсказуемым и стабильным, поскольку вероятность непредвиденных ситуаций и нестандартных ответов сводится к минимуму.

### Снижение риска ошибок и упрощение тестирования

При использовании аннотаций тестирование API становится проще, так как ответы для различных сценариев строго определены. Автоматизация тестов с аннотациями, такими как NotFoundResponse и InternalServerError, позволяет тестировщикам и разработчикам проверить каждый ответ на соответствие стандартам, заложенным в аннотациях. Это повышает уровень безопасности и устойчивости приложения, поскольку при изменениях в коде аннотации помогают удостовериться, что все возможные варианты ответов учтены и работают правильно. Кастомные аннотации облегчают интеграцию автоматических тестов и позволяют более гибко обрабатывать ошибки, не влияя на основную бизнес-логику приложения.

### Применение аннотаций API как стандарт разработки

Использование кастомных аннотаций для обработки ответов и ошибок становится стандартом разработки для многих приложений, особенно когда необходима гибкость и расширяемость. В приложении применение аннотаций позволяет не только обрабатывать ошибки и успешные ответы, но и закладывать основу для будущих улучшений. Если в дальнейшем будет необходимо добавить новые сценарии ответов или специфицировать обработку уникальных случаев, аннотации можно обновить и дополнить, сохраняя общее единообразие кода.

Таким образом, кастомные аннотации, такие как OkList, OkWord, NotFoundResponse, и InternalServerError, обеспечивают комплексное решение для управления ответами и обработкой ошибок в работе приложения. Они способствуют улучшению читаемости, поддерживаемости, стандартизации и документирования API, делая работу с приложением более удобной и предсказуемой.

## Заключение

Эта глава была посвящена разработке и архитектуре приложения Language Trainer и позволяет детально рассмотреть важные аспекты, которые формируют его функциональность и удобство использования. Приложение построено с использованием фреймворка Spring Boot, который предоставляет удобные инструменты для быстрой разработки и позволяет эффективно управлять данными, обеспечивая масштабируемость и простоту в обслуживании. Благодаря использованию Spring Data JPA и базы данных PostgreSQL удается гарантировать надежное хранение информации, в том числе данных о пользователях и их словарях. Это сочетание инструментов и технологий оптимально подходит для приложений, требующих хранения большого объема данных и высокой степени надежности.

Кроме того, особое внимание уделено безопасности и обработке ошибок. Система использует Spring Security для защиты данных пользователей, обеспечивая безопасную авторизацию и аутентификацию. Кастомные аннотации API и продуманные механизмы обработки ошибок позволяют унифицировать ответы сервера, что упрощает разработку, улучшает читаемость кода и повышает предсказуемость работы приложения. Это позволяет гибко управлять возможными исключениями, обрабатывать типовые ошибки и минимизировать негативные сценарии взаимодействия.

Архитектура MVC (Model-View-Controller), использованная в Language Trainer, чётко разделяет бизнес-логику, представления и взаимодействие с данными, что делает приложение легко расширяемым и удобным для пользователей. Применение Thymeleaf в качестве шаблонизатора упрощает разработку адаптивного пользовательского интерфейса, который поддерживает интерактивное взаимодействие с пользователем. Возможность сортировки слов и использование паттерна «Стратегия» для управления списками улучшают пользовательский опыт и делают работу с приложением более гибкой и удобной.

Все перечисленные решения позволяют Language Trainer эффективно выполнять свою основную задачу – быть удобным и функциональным тренажёром для изучения английского языка. Тщательный подбор технологий и продуманная архитектура дают возможность пользователям не только комфортно работать с приложением, но и полностью сосредоточиться на обучении, что делает Language Trainer мощным инструментом для практики и усвоения английской лексики.

# Практическая реализация приложения: архитектура, функциональность и интеграция технологий

Практическая часть разработки приложения Language Trainer направлена на создание удобного инструмента для тренировок по написанию английских слов, ориентированного на русскоязычных пользователей. Основные задачи включали реализацию функционала для регистрации и авторизации пользователей, управления списками слов, их проверки, сортировки, а также обеспечения надежного хранения данных и безопасности. В этой главе описаны ключевые этапы реализации, архитектурные решения и методы, которые использовались при создании приложения.

## Подготовка среды и конфигурация базы данных

Подготовка среды и конфигурация базы данных – один из первых и важных этапов создания приложения, поскольку он определяет, каким образом и где будут храниться все данные. В проекте была выбрана база данных PostgreSQL, которая зарекомендовала себя как мощное и надёжное решение для хранения больших объёмов информации. PostgreSQL предоставляет такие функции, как поддержка транзакций, индексов, различных типов данных и расширенных возможностей работы с JSON, что делает её подходящей для сложных приложений, требующих высокой производительности и безопасности данных. Этот выбор обусловлен тем, что приложение предполагает интенсивное взаимодействие с данными, включая обработку и хранение пользовательских аккаунтов, информации о словах, их переводах и статистике тренировок.

Для настройки подключения к базе данных в среде Spring Boot используется конфигурационный файл application.yaml, который предоставляет централизованное место для всех параметров поключения. В этом файле указаны такие ключевые параметры, как URL-адрес базы данных, а также учётные данные – логин и пароль для авторизации приложения в PostgreSQL. Эта конфигурация обеспечивает стабильное и безопасное подключение, минимизируя возможность ошибок при соединении и обеспечивая непрерывный доступ к базе данных.

Spring Boot в сочетании с Spring Data JPA значительно упрощает работу с базой данных благодаря использованию объектно-реляционного отображения (ORM). ORM позволяет разработчикам работать с сущностями базы данных как с объектами Java, что ускоряет разработку и снижает вероятность ошибок. Это означает, что приложение может взаимодействовать с таблицами, используя модели данных (сущности), что упрощает процесс программирования, так как для доступа к данным не требуется прямое написание SQL-запросов. Вместо этого Spring Data JPA генерирует SQL автоматически на основе операций, выполняемых над объектами сущностей, что также улучшает читаемость и поддержку кода.

Для автоматического создания и обновления таблиц в базе данных используется параметр hibernate.ddl-auto: update. Эта настройка позволяет Hibernate автоматически отслеживать изменения в структурах сущностей и вносить соответствующие обновления в базу данных. Если в процессе разработки сущность модели была изменена, Hibernate внесет нужные изменения в структуру базы, добавив новые поля, изменив типы данных или создав новые таблицы, если это необходимо. Такая автоматизация не только сокращает время на ручное внесение изменений, но и снижает риск несоответствия между кодом и структурой базы данных.

Благодаря этой настройке разработчики могут легко и быстро добавлять новые функции, требующие изменений в базе данных, без необходимости вручную изменять SQL-структуру таблиц. Это особенно полезно в условиях активной разработки, когда приложение может проходить через множество итераций и обновлений.

## Создание моделей данных

Создание моделей данных – один из ключевых аспектов построения приложения, поскольку модели данных определяют, как информация будет храниться и взаимодействовать с системой. В приложении разработаны основные модели данных, такие как User и Word, каждая из которых отвечает за хранение и представление специфических данных, необходимых для работы приложения. Эти модели описаны в виде Java-классов, что делает их лёгкими для работы и чтения, а использование аннотаций упрощает создание таблиц в базе данных и автоматизирует управление данными.

### Модель User

Модель User служит для хранения информации о пользователях, зарегистрированных в системе. Эта модель включает такие важные поля, как:

**Идентификатор пользователя (ID)**: уникальный идентификатор для каждого пользователя, аннотированный @Id и @GeneratedValue. Эти аннотации позволяют Spring Data JPA автоматически генерировать уникальные значения для каждого нового пользователя, что гарантирует, что все пользователи будут отличаться друг от друга в базе данных.

**Логин и пароль**: данные, необходимые для аутентификации и авторизации пользователя в системе. Поля логина и пароля могут быть дополнительно аннотированы для проверки их уникальности или обязательности заполнения, чтобы предотвращать ошибки и повысить безопасность данных. Аннотация @Column также может быть использована для указания, что логин должен быть уникальным, чтобы избежать дублирования пользователей.

Аннотации @Entity и @Table позволяют Spring автоматически распознавать класс User как сущность, а также преобразовывать его в таблицу в базе данных. Это означает, что, как только приложение запускается, Spring Data JPA создает таблицу User со всеми полями, указанными в классе, и все данные, относящиеся к пользователю, будут храниться в этой таблице. Это избавляет разработчиков от необходимости вручную создавать SQL-таблицы, что ускоряет процесс разработки.

### Модель Word

Модель Word предназначена для хранения информации о словах, с которыми пользователь взаимодействует в приложении. Данная модель включает следующие поля:

**Английское слово**: основное слово, которое необходимо запомнить и правильно написать. Это поле может быть аннотировано, например, @Column(nullable = false), что указывает на обязательность его заполнения.

**Русский перевод**: перевод слова на русский язык, который помогает пользователю легче запоминать значение.

**Дата последнего повторения**: поле, которое хранит дату последнего взаимодействия пользователя с данным словом. Это полезно для анализа и создания функционала, который возвращает пользователю те слова, которые он давно не повторял. Поле даты можно также использовать для автоматической сортировки слов по времени последнего повторения.

Эти данные помогают приложению более гибко взаимодействовать с пользователем, отслеживая его прогресс и анализируя, какие слова он выучил, а какие требуют больше практики. Аннотации, такие как @Entity и @Table, также используются для автоматического создания таблицы Word в базе данных, что позволяет системе эффективно сохранять и управлять данными о словах.

### Преимущества аннотаций и ORM в модели данных

Использование Spring Data JPA и аннотаций, таких как @Entity, @Table, @Id, и @GeneratedValue, позволяет избежать ручного написания SQL-запросов для создания таблиц и полей. Аннотации обеспечивают ORM (объектно-реляционное отображение) между классами Java и таблицами базы данных, что делает взаимодействие с базой данных более интуитивным и безопасным. Разработчики могут сосредоточиться на бизнес-логике приложения, в то время как ORM берет на себя всю рутину взаимодействия с базой данных. Например, добавление нового пользователя или слова в базу данных сводится к простому сохранению объекта в репозитории, а не к написанию SQL-запросов. Это также улучшает читаемость и поддерживаемость кода.

### Валидация и целостность данных

Для того чтобы сохранить целостность и корректность данных, в модели данных также предусмотрены аннотации для валидации. Например, можно использовать @NotNull, чтобы не допустить пустые значения в критически важных полях, таких как логин и пароль пользователя, или @Size, чтобы ограничить длину введенного текста. Валидация позволяет предотвратить ошибки на уровне данных, еще до их записи в базу, и делает взаимодействие с пользователем более удобным, предоставляя понятные сообщения о необходимости заполнения тех или иных полей.

Таким образом, модели User и Word в приложении обеспечивают надёжное хранение и управление данными пользователей и слов. Эти модели, описанные с помощью аннотаций, упрощают работу с базой данных, делают приложение более масштабируемым и улучшают структуру кода, благодаря чему весь процесс взаимодействия с данными становится быстрее и эффективнее.

## Архитектура MVC и создание контроллеров

Для реализации архитектурного подхода MVC (Model-View-Controller) в приложении были разработаны три ключевых компонента, каждый из которых отвечает за отдельный аспект логики и структуры приложения. Это позволяет добиться чёткого разделения ответственности, упростить разработку и поддержку кода, а также обеспечить масштабируемость и гибкость системы.

### Model: Определение моделей данных

Компонент **Model** представляет собой ядро приложения, отвечающее за хранение и управление данными. Основными моделями данных являются User и Word, которые представляют информацию о пользователях и их словарях соответственно:

**Модель User** включает такие поля, как идентификатор, имя пользователя и пароль, а также может содержать дополнительные атрибуты, необходимые для аутентификации, авторизации и управления учетной записью пользователя. Это помогает организовать хранение данных о пользователе и его идентификацию в системе.

**Модель Word** хранит информацию о словах и переводах, с которыми взаимодействует пользователь, а также дополнительную информацию, такую как дата последнего повторения слова. Это позволяет отслеживать прогресс пользователя и реализовывать функцию возврата редко используемых слов для повторения.

Обе модели аннотированы для автоматического преобразования их в таблицы базы данных с помощью ORM-инструментов Spring Data JPA, что упрощает управление данными и повышает скорость разработки.

### View: Представления на основе Thymeleaf

Компонент **View** отвечает за отображение данных и взаимодействие с пользователем. В приложении роль представлений выполняют шаблоны, созданные с помощью Thymeleaf – это мощный и удобный инструмент для генерации HTML-страниц:

Thymeleaf позволяет внедрять переменные и данные модели непосредственно в HTML-шаблоны, что делает страницы динамическими и интерактивными. Это значит, что отображение данных, например списка слов или информации о пользователе, становится более гибким и наглядным.

Представления в Language Trainer включают страницы для аутентификации, регистрации, работы сословарём, добавления и редактирования слов, а также для управления настройками пользователя. Это обеспечивает удобный интерфейс, через который пользователь может выполнять основные функции приложения.

Каждый шаблон в Thymeleaf связан с определённым контроллером, что позволяет контролировать представление данных и их обновление на стороне сервера, реагируя на действия пользователя.

### Controller: Управление запросами и взаимодействие с пользователем

Компонент **Controller** выполняет роль посредника между моделью и представлением, обрабатывая запросы от пользователя и выполняя соответствующие действия. Ключевыми контроллерами являются UserController и WordController:

**UserController** управляет функционалом, связанным с пользователем, включая регистрацию, аутентификацию, управление данными учётной записи и другие операции, связанные с профилем пользователя.

**WordController** реализует функции, связанные со словами и словарём пользователя. Он отвечает за отображение списка слов, добавление новых слов, редактирование и удаление существующих записей, а также за сортировку списка слов. Например, если пользователь запрашивает список слов, отсортированный по алфавиту или дате последнего повторения, контроллер определяет и применяет нужную стратегию сортировки, а затем передает данные в представление.

Контроллеры обрабатывают все пользовательские запросы и отправляют соответствующие данные модели во View. Например, при добавлении нового слова контроллер принимает данные, проверяет их корректность, добавляет в базу данных и обновляет список слов, представленный пользователю. В случае ошибки, контроллер возвращает сообщение об ошибке и предлагает повторить ввод.

### Интеграция компонентов MVC

В приложении, каждый компонент MVC тесно связан с остальными:

**Модели данных** предоставляют контроллерам доступ к информации, необходимой для выполнения действий, инициированных пользователем.

**Контроллеры** обрабатывают пользовательские запросы и в зависимости от их типа обращаются к моделям данных для выполнения операций, таких как добавление, обновление или удаление данных.

**Представления на базе Thymeleaf** получают данные от контроллеров и отображают их пользователю в удобной и понятной форме.

Такое разделение делает приложение более масштабируемым и лёгким для поддержки, поскольку разработчики могут изменять отдельные компоненты без влияния на остальную структуру. Например, можно доработать представления, добавив новые элементы интерфейса или изменить модели данных, не затрагивая работу контроллеров.

### Преимущества использования MVC

Применение MVC-подхода делает приложение удобным и гибким в развитии, поскольку каждая часть системы может развиваться независимо. Это также позволяет легко добавлять новый функционал. Например, при необходимости можно без труда добавить новые модели данных для учёта дополнительной информации о пользователе или его активности, подключить новые представления для расширенного интерфейса или добавить новый контроллер для управления новыми функциями.

В конечном итоге, MVC-структура обеспечивает структурированное разделение логики, что повышает читаемость и поддержку кода и создаёт мощную и удобную платформу для улучшения навыков пользователей в изучении английского языка.

## Реализация шаблонов с использованием Thymeleaf

Для создания динамических и удобных для пользователя страниц в приложении был выбран шаблонизатор **Thymeleaf**. Его основное преимущество заключается в тесной интеграции со Spring, что позволяет легко управлять данными и отображать их на HTML-страницах, создавая высокоадаптивный интерфейс. Благодаря поддержке различных возможностей для вставки данных, условного отображения элементов и использования циклов, Thymeleaf становится мощным инструментом для разработки веб-приложений.

### Основные представления приложения

В Language Trainer разработаны основные страницы, необходимые для взаимодействия пользователя с приложением:

**Страница списка слов**: представляет собой таблицу со словами пользователя. Здесь пользователь может видеть свой словарь, управлять словами – добавлять новые, редактировать или удалять уже существующие записи. С помощью Thymeleaf данные о словах, такие как английские и русские эквиваленты, даты последнего повторения и другие параметры, отображаются в виде таблицы с поддержкой сортировки по различным полям. При необходимости можно добавить и другие функции, такие как фильтрация по частоте использования или дате.

**Страница добавления слова**: предоставляет интерфейс для добавления нового слова в словарь пользователя. На странице отображаются поля ввода для английского слова и его русского перевода, а также кнопка для подтверждения ввода. С помощью Thymeleaf шаблон автоматически подгружает ошибочные сообщения или подтверждения успешного добавления, позволяя пользователю сразу видеть, было ли его действие успешным или нет.

**Страница проверки**: это основная учебная страница, на которой пользователю показывается русское слово, и он должен ввести его английский перевод. Thymeleaf помогает создать интерфейс, отображающий нужные переменные и обновляющийся по мере проверки ответов. После введения ответа пользователем система отправляет данные на сервер, проверяет их корректность и предоставляет пользователю немедленную обратную связь. При ошибке Thymeleaf отображает сообщение о неверном переводе и, при необходимости, дает подсказку для следующей попытки.

### Особенности использования Thymeleaf для внедрения данных

Thymeleaf позволяет внедрять переменные прямо в HTML-код, что делает шаблоны максимально адаптивными и простыми в использовании:

**Переменные**: Thymeleaf позволяет вставлять значения переменных Spring, таких как имя пользователя, текущий счёт или дата последнего повторения, прямо в HTML-страницы. Это особенно полезно на странице списка слов, где можно динамически подгружать информацию о каждом слове.

**Условные выражения**: Thymeleaf поддерживает условные конструкции, позволяющие управлять отображением элементов. Например, можно задать условие, чтобы при отсутствии слов в словаре вместо таблицы с данными отображалось сообщение «Ваш словарь пуст».

**Циклы**: Thymeleaf позволяет легко отображать списки данных, используя цикл th:each. На странице списка слов это используется для отображения каждой записи из словаря пользователя, делая шаблон динамическим и гибким.

**Обработка ошибок и сообщений**: Thymeleaf позволяет автоматически отображать сообщения о результатах операции. Например, на странице добавления слова можно встроить проверку, чтобы после добавления нового слова выводилось сообщение об успешной операции, а при ошибке – соответствующее уведомление.

### Преимущества использования Thymeleaf для создания адаптивного интерфейса

Использование Thymeleaf упрощает процесс создания удобного и адаптивного интерфейса для пользователя. Благодаря поддержке вставки данных, условного отображения и работы с циклами, Thymeleaf позволяет избежать сложного JavaScript-кода на фронтенде и обеспечить максимальную интерактивность за счет Spring:

Thymeleaf позволяет обновлять данные на сервере и немедленно отображать их на клиентской стороне, что ускоряет интерфейс и делает его удобнее.

Гибкость в управлении данными через Thymeleaf позволяет добавлять новые функции – от поиска и фильтрации слов до отображения прогресса в изучении языка.

Thymeleaf, являясь шаблонизатором, интегрированным со Spring, предоставляет богатый функционал для создания высокоадаптивного и удобного интерфейса. Это позволило создать страницы, обеспечивающие лёгкое добавление новых функций, таких как сортировка, фильтрация и проверка переводов, без необходимости в сложном JavaScript-коде.

### Реализация функций сортировки и использования паттерна «Стратегия»

Для повышения удобства работы с приложением была реализована гибкая система сортировки списка слов, предоставляющая пользователю возможность выбирать порядок отображения по различным критериям. Основная цель – создать настраиваемую сортировку, позволяющую пользователю сортировать слова как по английским терминам, так и по русским. Для реализации этой функции был использован паттерн проектирования «Стратегия», который позволяет легко переключаться между различными алгоритмами сортировки, сохраняя модульность и расширяемость кода.

### Применение паттерна «Стратегия» для сортировки данных

Паттерн «Стратегия» был выбран для реализации данной функции, так как он позволяет определять семейство алгоритмов, инкапсулировать каждый из них и обеспечить взаимозаменяемость. Это значит, что можно легко добавлять новые стратегии сортировки или изменять существующие, не нарушая остальную логику работы приложения. В рамках этого подхода были созданы два класса, EnglishSortStrategy и RussianSortStrategy, каждый из которых представляет отдельную стратегию сортировки.

EnglishSortStrategy – отвечает за сортировку списка слов по английским терминам. В классе реализован метод, который сортирует данные в алфавитном порядке, основываясь на английских словах в базе данных.

RussianSortStrategy – реализует сортировку по русским эквивалентам слов. Этот класс позволяет пользователю видеть список отсортированным в порядке, зависящем от русских переводов, что может быть полезно для пользователей, ориентирующихся на родной язык при обучении.

### Механизм выбора и применения стратегий сортировки

В приложении создан класс WordService, который взаимодействует со стратегиями и отвечает за применение нужного алгоритма сортировки в зависимости от предпочтений пользователя. Механизм работы следующий:

Когда пользователь выбирает предпочтительный способ сортировки (по английским или русским словам), WordService принимает это предпочтение и передаёт его выбранной стратегии.

В WordService реализована логика, которая динамически определяет, какая именно стратегия должна быть использована, и применяет её метод сортировки к данным.

Такая структура делает приложение гибким и поддерживаемым, позволяя в будущем добавлять новые критерии сортировки, такие как сортировка по дате последнего повторения или по частоте использования, с минимальными изменениями в коде.

### Преимущества использования паттерна «Стратегия» для сортировки

Использование паттерна «Стратегия» позволяет значительно улучшить модульность и гибкость приложения, а также упрощает добавление новых функций. Вот некоторые ключевые преимущества:

Легкость масштабирования. Если потребуется добавить новые критерии сортировки (например, по дате или частоте использования), достаточно будет создать новый класс с реализацией соответствующего метода сортировки.

Поддерживаемость кода. Паттерн «Стратегия» изолирует каждую стратегию сортировки в отдельном классе, что облегчает модификацию и тестирование алгоритмов. Внесение изменений в один алгоритм не влияет на работу других.

Удобство для пользователя. Пользователи могут выбирать наиболее удобный для них способ сортировки, что повышает комфорт работы с приложением и позволяет настроить интерфейс под индивидуальные предпочтения.

Реализация сортировки с использованием паттерна «Стратегия» сделала Language Trainer более адаптивным и ориентированным на пользователя. Благодаря этому подходу приложение предоставляет пользователям возможность гибко управлять отображением своего словаря, выбирая наиболее подходящий метод сортировки. Кроме того, данный подход упростил поддержку и расширяемость кода, поскольку добавление новых алгоритмов сортировки теперь требует минимальных изменений. Таким образом, паттерн «Стратегия» не только улучшил функциональность, но и значительно повысил удобство использования и гибкость системы.

### Аутентификация и авторизация с использованием Spring Security

Для защиты пользовательских данных и обеспечения надежности доступа приложение использует Spring Security – мощный инструмент, который предоставляет различные функции для управления безопасностью на уровне приложения. Spring Security интегрируется в структуру MVC, применяя дополнительные фильтры и компоненты для управления доступом к ресурсам и предотвращения несанкционированного использования системы.

Основные принципы работы Spring Security

В рамкахприложения реализованы как аутентификация, так и авторизация, которые обеспечивают контроль над доступом к пользовательским данным и различным функциям приложения. Аутентификация идентифицирует пользователя и проверяет его учётные данные, позволяя убедиться, что он действительно является тем, за кого себя выдает. Авторизация, в свою очередь, проверяет, какие действия разрешены пользователю, и ограничивает доступ к определённым ресурсам для неавторизованных пользователей. Это разделение позволяет установить чёткие уровни доступа, где только авторизованные пользователи имеют возможность добавлять и изменять свои списки слов или редактировать свои данные, а неавторизованные пользователи могут просматривать лишь ограниченную информацию, как, например, страницу входа и регистрации.

**Конфигурация и настройка Spring Security**

Spring Security поддерживает гибкую конфигурацию, что позволяет адаптировать его под потребности приложения. В определены наборы правил, которые контролируют доступ к различным маршрутам и ограничивают функциональность для неавторизованных пользователей. Например, для доступа к страницам редактирования или управления словарём требуется авторизация, в то время как страница регистрации доступна всем пользователям. Эта настройка безопасности осуществляется с использованием аннотаций и конфигурационных классов, которые определяют, какие URL защищены, и какие методы аутентификации требуются.

**Работа с сессиями и безопасностью паролей**

Одним из преимуществ Spring Security является встроенная поддержка сессий, что позволяет отслеживать авторизованных пользователей в течение их работы с приложением. После успешного входа в систему пользователю создается сессия, которая сохраняет его аутентификационные данные, обеспечивая постоянство авторизации до завершения сеанса или выхода. Это обеспечивает удобство работы, так как пользователю не нужно повторно проходить аутентификацию на каждой странице.

Также важным аспектом является защита паролей. В приложении пароли пользователей хранятся в зашифрованном виде с использованием надёжных алгоритмов хэширования. Spring Security автоматически управляет процессом шифрования и дешифрования паролей, предоставляя для этого такие классы, как BCryptPasswordEncoder. Этот класс применяется для хеширования паролей при их сохранении в базе данных и при проверке введённого пароля на этапе аутентификации. Благодаря этому подходу защита данных становится более устойчивой к атакам и обеспечивает пользователям безопасность хранения их личных данных.

**Интеграция с формами и обработка ошибок аутентификации**

Spring Security интегрируется с веб-формами для входа и регистрации, которые отображаются пользователю при необходимости. Приложение поддерживает формы, обрабатывающие вход и регистрацию, а в случае неудачной попытки входа, например при неправильном вводе пароля или логина, пользователю отображается уведомление об ошибке. Spring Security позволяет гибко настраивать сообщения об ошибках и перенаправления, предоставляя пользователю полезную обратную связь и обеспечивая удобное взаимодействие.

Преимущества использования Spring Security в Language Trainer

Использование Spring Security предоставляет ряд преимуществ: приложение получает централизованную систему защиты, поддерживающую современные стандарты безопасности, такие как шифрование данных, защита сессий и интеграция с пользовательскими интерфейсами. Также, благодаря автоматизации значительной части процессов, связанных с безопасностью, такие как аутентификация и авторизация, снижается нагрузка на разработчиков, что позволяет сосредоточиться на развитии основной функциональности.

### Валидация данных и обработка ошибок

Валидация данных и корректная обработка ошибок играют ключевую роль в повышении устойчивости и удобства использования. Учитывая, что пользователи регулярно добавляют и проверяют слова в своём словаре, приложение должно гарантировать целостность данных и предотвращать ошибки ввода. Это позволяет избегать ситуаций, когда некорректные или пустые значения приводят к непредсказуемому поведению приложения или сбоям в работе.

**Валидация данных при добавлении и проверке слов**

На этапе добавления нового слова в словарь реализована многоуровневая проверка, которая предотвращает сохранение неполных данных. Например, при добавлении слова пользователю необходимо ввести как английский эквивалент, так и его перевод на русском языке. Если одно из полей остается пустым или содержит лишь пробелы, пользователь получает уведомление о необходимости заполнения обоих полей. Такая валидация не только предотвращает возможные ошибки, но и улучшает удобство использования, помогая пользователю вводить корректные данные.

Кроме того, валидация проводится и на уровне сервера. Это позволяет избежать случайного добавления пустых записей даже в тех случаях, когда клиентская валидация по какой-то причине не сработала. Использование серверных аннотаций, таких как @NotNull и @Size, помогает гарантировать, что при сохранении данных в базе каждый элемент будет соответствовать заданным требованиям.

**Отображение сообщений об ошибках для пользователя**

Когда приложение обнаруживает ошибку в пользовательском вводе, оно не просто блокирует действие, но и предоставляет пользователю полезное сообщение с объяснением причины. Например, если пользователь не заполнил одно из обязательных полей, приложение выводит сообщение о необходимости ввода данных. Это помогает избежать потенциальной фрустрации пользователя и упрощает работу с приложением, предоставляя чёткие рекомендации по исправлению ошибок.

**Обработка ошибок, связанных с базой данных**

Работа с базой данных также требует тщательно продуманного механизма обработки ошибок. Например, при попытке доступа к записи, которой нет в базе данных, может возникнуть ошибка, если приложение попытается извлечь несуществующую информацию. Чтобы предотвратить подобные сбои, реализовано исключение NotFoundWordException. Данное исключение вызывается, когда приложение не находит нужное слово по запросу пользователя, и вместо технической ошибки пользователь получает страницу с информативным сообщением о том, что запрашиваемое слово не найдено. Это делает взаимодействие с приложением более понятным и управляемым, так как пользователь всегда получает обратную связь о статусе своего запроса.

**Преимущества централизованной обработки ошибок**

Благодаря централизованной системе обработки ошибок в приложении стало проще управлять различными сценариями сбоев. Такой подход обеспечивает единообразие в отображении ошибок, позволяя предоставить пользователям понятные сообщения, которые объясняют, что именно произошло и как можно исправить ситуацию. Использование аннотаций, специализированных исключений и стандартных ответов на ошибки также делает код более поддерживаемым и облегчает его обновление. В случае добавления новых функций, связанных с валидацией данных или взаимодействием с базой, система обработки ошибок легко расширяется, поддерживая общую структуру приложения.

Валидация и обработка ошибок как часть пользовательского опыта

Функции валидации и обработки ошибок – это не просто меры безопасности, но и важная часть пользовательского опыта. Грамотно реализованная система уведомлений помогает пользователю чувствовать себя уверенно, делая его взаимодействие с приложением более прозрачным и удобным. В случае ошибки он получает чёткую инструкцию, как поступить дальше, что повышает доверие к приложению и стимулирует его дальнейшее использование.

### Документирование API с помощью аннотаций OpenAPI

В проекте для документирования API были использованы аннотации OpenAPI, что позволило стандартизировать описание возможных ответов и улучшить читаемость кода. Внедрение кастомных аннотаций, таких как @OkList, @OkWord, @NotFoundResponse и @InternalServerError, поможет быстро ориентироваться в структуре приложения и понимать, какие типы ответов следует ожидать от различных методов контроллеров.

Аннотации @OkList и @OkWord обеспечивают описание успешных ответов, возвращаемых контроллерами, и дают возможность другим разработчикам легко понять, в каком формате будет предоставлен ответ. Например, аннотация @OkList используется для указания, что метод возвращает список элементов, таких как слова или пользователи, а @OkWord – для возврата конкретного слова. Эти аннотации не только упрощают понимание структуры возвращаемых данных, но и помогают автоматизировать тестирование и интеграцию API.

Ошибки также документированы с помощью @NotFoundResponse и @InternalServerError. @NotFoundResponse используется для обработки случаев, когда запрашиваемый ресурс (например, слово) не найден в базе данных. Это позволяет другим разработчикам и пользователям API сразу видеть, что данный тип ошибки может возникнуть, и обрабатывать его соответствующим образом. @InternalServerError сигнализирует о возможных непредвиденных ошибках на стороне сервера и дает пользователям API понять, что при возникновении этой ошибки необходимо выполнить дополнительные действия для диагностики или связаться с разработчиком.

Эти аннотации также позволяют легко генерировать документацию API в формате OpenAPI (например, с использованием Swagger), что автоматически создаёт интерфейс для тестирования и использования API. Это особенно полезно для крупных проектов, где присутствует множество методов и различных типов ответов. Благодаря этому, пользователи API могут сразу ознакомиться с его возможностями, сценариями ответов и потенциальными ошибками, не изучая каждую строчку кода.

### Заключение

Практическая часть разработки приложения показала, как важны современные технологии, методы и паттерны для создания эффективного и устойчивого инструмента для языкового обучения. Использование таких компонентов, как Spring Boot, Spring Security и Thymeleaf, позволило не только обеспечить высокую производительность и стабильность приложения, но и сделать его удобным и безопасным для пользователей, стремящихся улучшить свои навыки в изучении английского языка.

С помощью Spring Boot была значительно упрощена настройка и конфигурация приложения: этот фреймворк позволил избежать сложностей при подключении к базе данных PostgreSQL и автоматизировал процесс работы с объектно-реляционным отображением через Spring Data JPA. Благодаря этому удалось сосредоточиться на основной бизнес-логике приложения, а не на трудоёмкой настройке инфраструктуры. Spring Security добавил важный уровень защиты данных пользователей, обеспечив надёжную аутентификацию и авторизацию. Это особенно важно в приложениях, связанных с языковым обучением, где доступ к персональным данным должен быть строго ограничен и защищён.

Реализация гибкого пользовательского интерфейса с помощью Thymeleaf позволила упростить взаимодействие с пользователем, создавая динамические страницы для регистрации, проверки и управления словарём. Внедрение паттерна «Стратегия» для сортировки списка слов дало возможность разделить логику сортировки и легко адаптировать приложение под требования пользователя, не усложняя структуру кода. Этот подход повысил гибкость приложения, позволив пользователю выбирать предпочтительный способ отображения слов – по английскому или русскому языку.

Также стоит отметить использование кастомных аннотаций API и интеграцию с OpenAPI для документирования всех возможных ответов API. Это дало разработчикам и пользователям возможность понимать и предугадывать сценарии взаимодействия с приложением, упрощая процесс разработки и тестирования. Аннотации, такие как @OkList, @OkWord, @NotFoundResponse и @InternalServerError, помогли структурировать код, улучшая его читаемость и доступность. Автоматически сгенерированная документация с помощью OpenAPI позволила внешним пользователям API быстрее и проще начать работу с приложением, минимизируя количество потенциальных ошибок и упрощая отладку.

В результате, Language Trainer стал не только тренажёром для написания английских слов, но и полноценным, структурированным и поддерживаемым продуктом, отвечающим требованиям современных пользователей. Внедрение передовых технологий и структурных решений сделало его надежным, масштабируемым и способным к дальнейшему расширению, а пользовательский интерфейс и функционал обеспечили удобство и комфорт для тех, кто стремится к улучшению своих языковых навыков.

# Заключение

В ходе выполнения дипломной работы были достигнуты как теоретические, так и практические результаты, подтверждающие актуальность и значимость разработки приложений для языкового обучения. Теоретический анализ существующих языковых тренажёров позволил выявить особенности эффективных методик изучения иностранных языков, таких как регулярные тренировки, фокус на написании и развитие навыков запоминания и воспроизведения слов. Это исследование обосновало выбор ключевых компонентов проекта, включая архитектуру MVC, применение Java Spring и других современных инструментов, обеспечивших удобство и безопасность работы с данными пользователя.

В рамках практического исследования и разработки удалось создать функциональное приложение, которое помогает пользователям тренировать навыки написания английских слов, предлагая широкий набор возможностей для гибкого взаимодействия с собственным словрём. Использование паттерна «Стратегия» для сортировки, интеграция с Thymeleaf для динамического интерфейса и защита данных через Spring Security позволили сделать работу с приложением надежной и интуитивно понятной.

### Оценка проведённого исследования и его результатов

Разработанное приложение продемонстрировало высокую стабильность и функциональность. Достигнута цель исследования – создание адаптивного и удобного инструмента для тренировки навыков написания, обеспечивающего безопасное и надёжное хранение данных. Задачи, поставленные в начале работы, были успешно решены, включая внедрение механизмов сортировки, обработки ошибок и валидации данных. Проведенное исследование подтвердило гипотезу о том, что использование современных фреймворков и паттернов проектирования делает приложение более структурированным и гибким, облегчая его дальнейшее сопровождение и развитие.

### Практическая значимость и рекомендации

Практическая значимость работы заключается в создании эффективного учебного инструмента, который может быть полезен как самостоятельным пользователям, так и образовательным учреждениям. Language Trainer может легко адаптироваться для обучения другим языкам или быть расширен для интеграции с другими образовательными платформами, что делает его удобным и востребованным приложением в сфере цифрового обучения. Рекомендуется в будущем расширить функциональность за счёт добавления поддержки других языков, создания более гибких настроек уровня сложности заданий и, возможно, внедрения системы рекомендаций для адаптации словаря под интересы пользователя.

### Общий итог и планы на дальнейшее совершенствование

Выполненная работа продемонстрировала, что цель исследования достигнута: разработано надёжное, интуитивно понятное и функциональное приложение для языкового обучения. Реализованные задачи и исследование подтвердили выбранные теоретические и технологические решения, доказав их значимость для создания качественного учебного ПО. В перспективе объект исследования можно улучшить путём добавления новых функций, таких как адаптивные тесты, а также интеграция с мобильными платформами для повышения доступности и гибкости использования.

# Список используемой литературы

1. *Блинов, А. В. Основы программирования на языке Java. Москва: Издательство Питер, 2019. 450 с.*
2. *Иванов, В. А., и Петров, И. И. Программные шаблоны проектирования. Москва: Издательство ДМК, 2020. 380 с.*
3. *Нестеренко, С. В. Веб-программирование на языке Java. Санкт-Петербург: БХВ-Петербург, 2021. 315 с.*
4. *Хорстманн, К. С., Корнелл, Г. Java 2. Основы. Москва: Вильямс, 2018. 850 с.*
5. *Fowler, M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2002. 560 p.*
6. *Johnson, R. E., и др. Spring Framework Documentation. San Francisco: Pivotal Software, 2021. 900 p.*
7. *Laddad, R. AspectJ in Action: Enterprise AOP with Spring. Greenwich: Manning, 2019. 550 p.*
8. *Walls, C. Spring in Action. Shelter Island: Manning Publications, 2021. 640 p.*
9. *Spring Framework Documentation. "Spring Security Overview." Доступно по ссылке:*[*https://spring.io/projects/spring-security*](https://spring.io/projects/spring-security)*.*
10. *PostgreSQL Documentation. "Introduction to PostgreSQL." Доступно по ссылке:*[*https://www.postgresql.org/docs/*](https://www.postgresql.org/docs/)*.*
11. *Thymeleaf Documentation. "Thymeleaf Templates and HTML Integration." Доступно по ссылке:*[*https://www.thymeleaf.org/*](https://www.thymeleaf.org/)*.*
12. *Oracle Java Documentation. "Java Platform, Standard Edition Documentation." Доступно по ссылке:*[*https://docs.oracle.com/javase/*](https://docs.oracle.com/javase/)*.*
13. *OpenAPI Documentation. "OpenAPI Specification Documentation." Доступно по ссылке: https://swagger.io/specification*