Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Мобильное приложение для изучения японского языка»

Студент ИУ7-61Б (Группа) _____ (Подпись, дата) Корниенко К. Ю. (И. О. Фамилия)

Руководитель курсовой работы _____ (Подпись, дата) Шибанова Д. А. (И. О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка 38 с., 10 рис., 2 табл., 22 источн., 1 прил.

Цель работы: разработка мобильного приложения, предназначенного для упрощения обучения японскому языку.

Мобильное приложение для изучения японского языка является удобным и эффективным инструментом для тех, кто хочет освоить язык. Его интерактивные функции и доступность на мобильных устройствах делают процесс изучения японского языка более гибким и увлекательным. Разработка такого приложения имеет потенциал для применения в учебных заведениях и самостоятельного обучения.

В разрабатываемом программном обеспечении выделены три роли: пользователь, модератор и администратор. Администратор добавляет новых пользователей и изменяет данные о них. Модератор вносит тексты на оптическое распознавание, а также дает доступ к текстам другим пользователям. Пользователи могут читать тексты, получать перевод и чтение слов и добавлять их в словарь.

Программа разработана на языке Dart с использованием фреймворка Flutter, что дает возможность разрабатывать приложение, поддерживаемое на нескольких платформах, с использованием единой кодовой базы.

Ключевые слова: оптическое распознавание иероглифов, база данных, система управления базами данных, мобильное приложение.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениями.

Сетевое обучение — парадигма учебной деятельности, основой которой является открытость образовательных ресурсов, а также повсеместное сотрудничество участников образовательного процесса.

Кана — японские слоговые алфавиты: хирагана и катакана.

Хирагана — японский слоговой алфавит, который используется для записи слов, не имеющих кандзи.

Катакана — японский слоговой алфавит, который используется для записи иностранных слов и выделения важных слов в тексте.

Кандзи — китайские иероглифы, которые используются в японской письменности. Кандзи были заимствованы из китайского языка и адаптированы для использования в японском языке

ВВЕДЕНИЕ

В последние годы распространена идея использования цифровых технологий для помощи студентам в преодолении трудностей, связанных с изучением японского языка, таких как сложная система письма, грамматика и синтаксис, а также необходимость запоминания большого количества новых слов и символов [1]. Также рассматривается применение сетевого обучения, как парадигмы в дистанционном обучении для улучшения взаимодействия между учащимися путем создания виртуальных общностей учащихся [2].

Одной из наиболее важных проблем, с которым сталкивается человек при изучении японского языка — сложная система письма которая включает в себя два алфавита и иероглифику. Оптическое распознавание символов может облегчить изучение японского языка, преобразуя изображения, содержащие японский язык, в текстовые документы, для более легкой обработки, перевода и изучения иероглифов [3].

Цель работы — разработать мобильное приложение для упрощения изучения японского языка.

Для достижения поставленной цели, необходимо выполнить следующие задачи:

- провести анализ предметной области;
- описать взаимодействие компонентов системы и спроектировать базу данных;
- описать интерфейс доступа к базе данных и произвести тестирование функционала;
- исследовать характеристики разработанного программного обеспечения.

1 Аналитический раздел

1.1 Использование цифровых технологий в изучении языков

Использование цифровых технологий в изучении японского языка является актуальным и эффективным подходом для изучения японского языка и повышения мотивации в процессе изучения [4]. Проблемы, с которыми сталкиваются изучающие японский язык, включают необходимость запоминания большого количества новых слов и иероглифов, а также сложности в понимании контекста и культурных отличий.

Во время круглого стола по технологиям для усточнивого развития, организованного ЮНЕСКО в Париже в 2013 г., проведенные исследования и изученные отчеты, выделяющие значительное преимущество применения информационных технологий в изучении языков и в преподавании, позволили сделать вывод о главенствующей роли цифровизации в формировании коллективных знаний [5].

1.2 Базы данных, системы управления базами данных

База данных (БД) — это компьютеризированная система, основное назначение которой — хранить информацию, предоставляя пользователям средства ее извлечения и модификации [6, с. 46]. Базы данных используются для различных целей, таких как управление бизнесом, научные исследования и медицинские записи.

Работа напрямую с базой данных может привести к нарушению целостности данных. Код для работы с базой данных сложен и неудобен, что может привести к ошибкам и проблемам с безопасностью. Избежать проблемы при работе с данными позволяет система управления базами данных.

Система управления базами данных (СУБД) - это программное обеспечение, которое позволяет пользователям создавать, управлять и обрабатывать данные в БД [7, с. 10—12]. СУБД предоставляет интерфейс для работы с данными, а также обеспечивает безопасность и целостность данных.

СУБД и БД тесно связаны, поскольку системы управления базами данных обеспечивают доступ к данным, хранящимся в БД, и позволяют

пользователям выполнять операции с данными.

1.3 Выбор системы управления базами данных

Для разработки мобильного приложения, которое будет преобразовывать фотографии текстов на японском языке в текстовые документы, может быть использована как реляционная система управления базами данных, так и нереляционная. При разработке приложения будет использована реляционная система управления базами данных, так как она позволяет оформить ролевую модель и хранить структурированные данные [8].

Реляционные системы управления базами данных характеризуются использованием реляционной модели управления, отличающуюся табличной формой представления данных, а также применением формальной математики и реляционных вычислений для обрабатываемых данных [9].

Данные, в реляционных моделях, представляют собой двумерный массив и характеризуются следующими особенностями:

- любая составляющая таблицы является одной составляющей данных;
- любой столбец имеет свое уникальное имя;
- отсутствие одинаковых строк в таблице;
- все составляющие в столбцах имеют однородный тип;
- строки и столбцы имеют произвольный порядок [10].

Основные современные СУБД основаны на реляционной модели данных, в таблице 1.1 представлен сравнительный анализ некоторых из них.

Таблица 1.1 – Сравнительный анализ реляционных СУБД

СУБД	Лицензия	Масштабируемость	Скорость выполнения запросов	Управление транзакциями
MSSQL [11]	проприетарная	вертикальная, комплексная горизонтальная	средняя	пессимистический
MariaDB [12]	GNU GPL	вертикальная горизонтальная	средняя	частично-оптимистический
PostgreSQL [13]	открытый исходный код	вертикальная горизонтальная	высокая	частично-оптимистический
Oracle [14]	проприетарная	вертикальная горизонтальная	высокая	оптимистическое
IBM DB2 [15]	проприетарная	вертикальная горизонтальная	средняя	пессимистический
SQLite [16]	MIT	вертикальная	низкая	частично-пессимистический

PostgreSQL обладает высокой производительностью и безопасностью, а также является бесплатным программным продуктом с открытым исходным кодом [10], что делает его доступным для использования при разработке мобильного приложения. Кроме того, PostgreSQL имеет хорошую поддержку хранимых процедур и триггеров, имеет хорошую поддержку для языков SQL и Unicode, что важно для работы с японским языком. Например, PostgreSQL поддерживает полнотекстовый поиск на японском языке и имеет встроенную поддержку для японских иероглифов [13], что делает его хорошим выбором для разрабатываемого приложения.

1.4 Обзор существующего программного обеспечения для упрощения изучения японского языка

При изучении иностранных языков часто используются мобильные приложения, изучение языков с поддержкой мобильных устройств позволяет обучающемуся получить доступ к знаниям о грамматике и лексике иностран-

ного языка, не накладывая ограничений на место и время изучения [17]. Особенно информационные технологии важны при изучении японского языка, который отличается двумя азбуками и иероглификой, ведь система письменности является важным аспектом при обучении любому иностранному языку.

В таблице 1.2 представлен сравнительный анализ четырех приложений для изучения японского языка.

Таблица 1.2 – Обзор приложений для упрощения изучения японского языка

ПО	Duolingo [18]	Memrise [19]	Lingodeer [20]	WaniKani [21]
Цена	Бесплатно с платными функциями	Бесплатно с платными функциями	Бесплатно с платными функциями	платно
Типы упражнений	перевод, аудирование, грамматика	перевод, аудирование, грамматика	перевод, аудирование, грамматика	кана, кандзи, слова, грамматика
Платформа	веб, мобильное приложение	веб, мобильное приложение	веб, мобильное приложение	веб, мобильное приложение

Исходя из приведенных в таблице данных, можно сделать вывод о том, что приложения не уделяют достаточно внимания чтению текстов на японском языке. В то же время именно чтение текстов на японском языке представляет наибольшую сложность в процессе обучения [22]. Для упрощения изучения японского языка в разрабатываемом мобильном приложении будет сделан акцент на снижение сложности чтения и анализа иероглифических текстов путем синтеза текстовых документов из изображений отрывков из книг, журналов и других источников текстов на японском языке.

1.5 Проектирование базы данных

1.5.1 Диаграмма базы данных в нотации Чена

Для проектируемой базы данных создана ER-диаграмма в нотации Чена, представленная на рисунке 1.1.

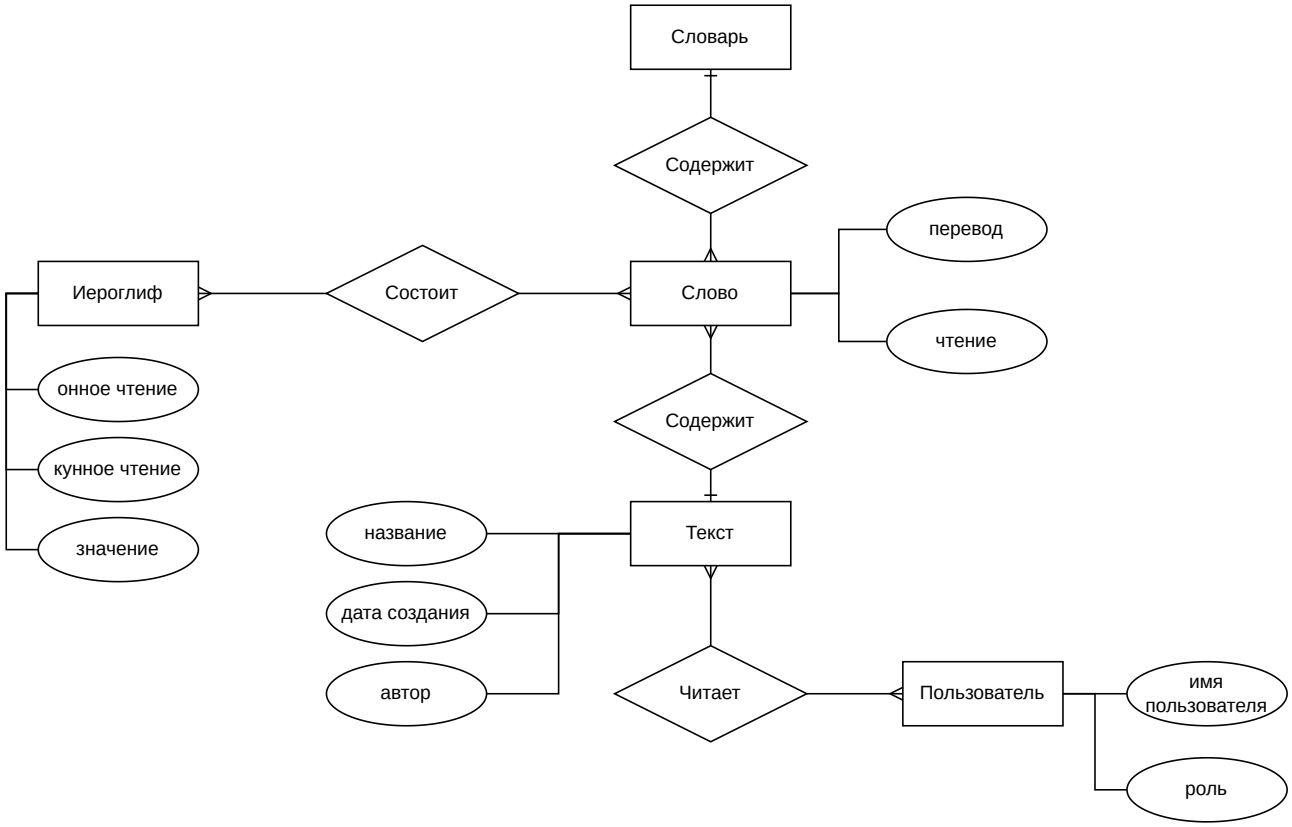

Рисунок 1.1 – Диаграмма сущность-связь в нотации Чена

Были выделенны 5 сущностей: *иероглиф*, *слово*, *текст*, *пользователь* и *словарь*. Отношения «многие-ко-многим» выделены между сущностями иероглиф и слово, текст и пользователь.

1.5.2 Пользовательские роли проектируемого приложения

В разрабатываемом приложении выделяются три роли: администратор, модератор и пользователь.

1. Администратор — имеет полный доступ к приложению и базе данных. Он может создавать и удалять учетные записи пользователей и модераторов, а также имеет доступ к полной информации о пользователях и их активности в приложении.

2. Модератор — имеет ограниченный доступ к приложению и базе данных. Он может создавать тексты и выдавать доступ к ним для пользователей. Модератор не имеет доступа к полной информации о пользователях и их активности в приложении.

3. Пользователь — имеет доступ к основным функциям приложения, таким как изучение японского языка, создание своего профиля. Пользователь не имеет полного доступа к базе данных.

Описание пользовательских сценариев представлено на рисунке 1.2.

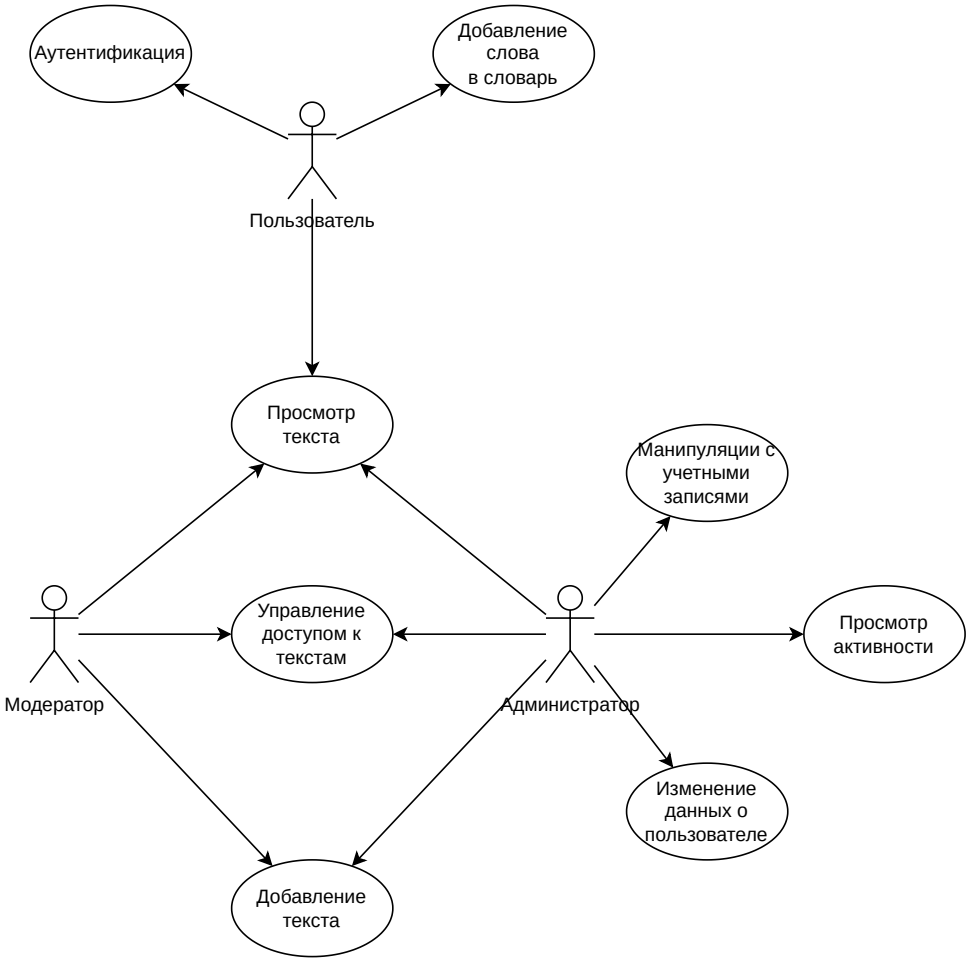

Рисунок 1.2 – Диаграмма вариантов использования приложения

Вывод из аналитического раздела

В данном разделе была рассмотрена предметная область и подходы к изучению японского языка. Также были представлены основные сведения о базах данных и системах управления базами данных для хранения японских иероглифов и проведен анализ существующих приложений для

упрощения изучения японского языка. В ходе работы будет использоваться СУБД PostgreSQL, так как она имеет открытый исходный код, высокую производительность и безопасность, а также поддерживает хранение в базе данных иероглифов, процедур и триггеров.

При разработке приложения будет учитываться отсутствие у крупных приложений для изучения японского языка механизмов для чтения печатных текстов. Приложение будет доступно как в веб версии, так и на мобильных платформах и персональных компьютерах.

2 Конструкторский раздел

2.1 Концептуальная модель системы

Фундаментальной функцией разрабатываемого приложения является преобразование изображения, содержащего текст, написанный на японском языке, в текстовый документ. Процесс преобразования изображения в текст состоит из нескольких этапов:

— проверка формата изображения;

— загрузка изображения для обработки;

— форматирование результата.

На рисунках 2.1–2.2 представлена модель системы в нотации IDEF0.

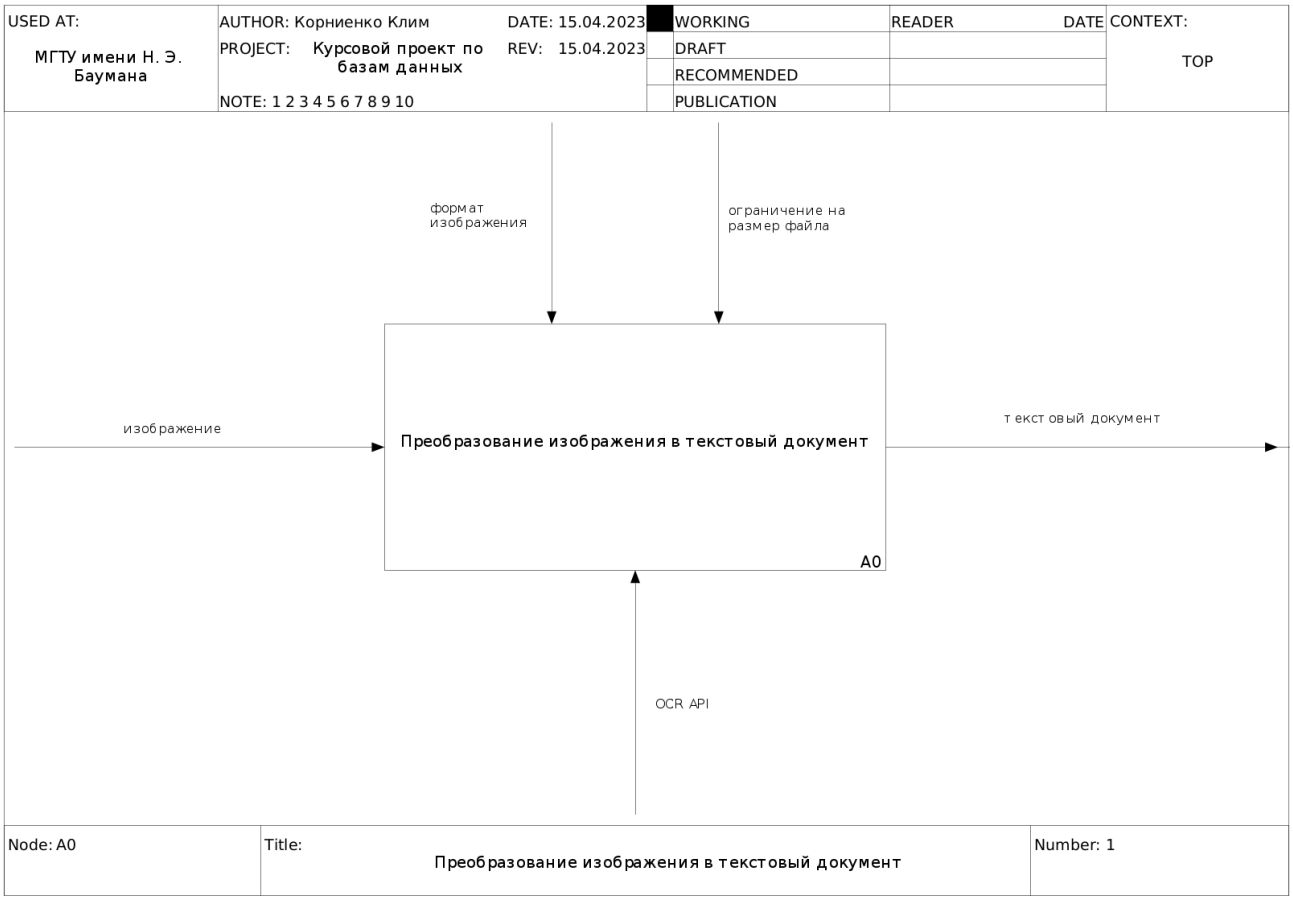

Рисунок 2.1 – Модель преобразования изображения в текст (верхний уровень)

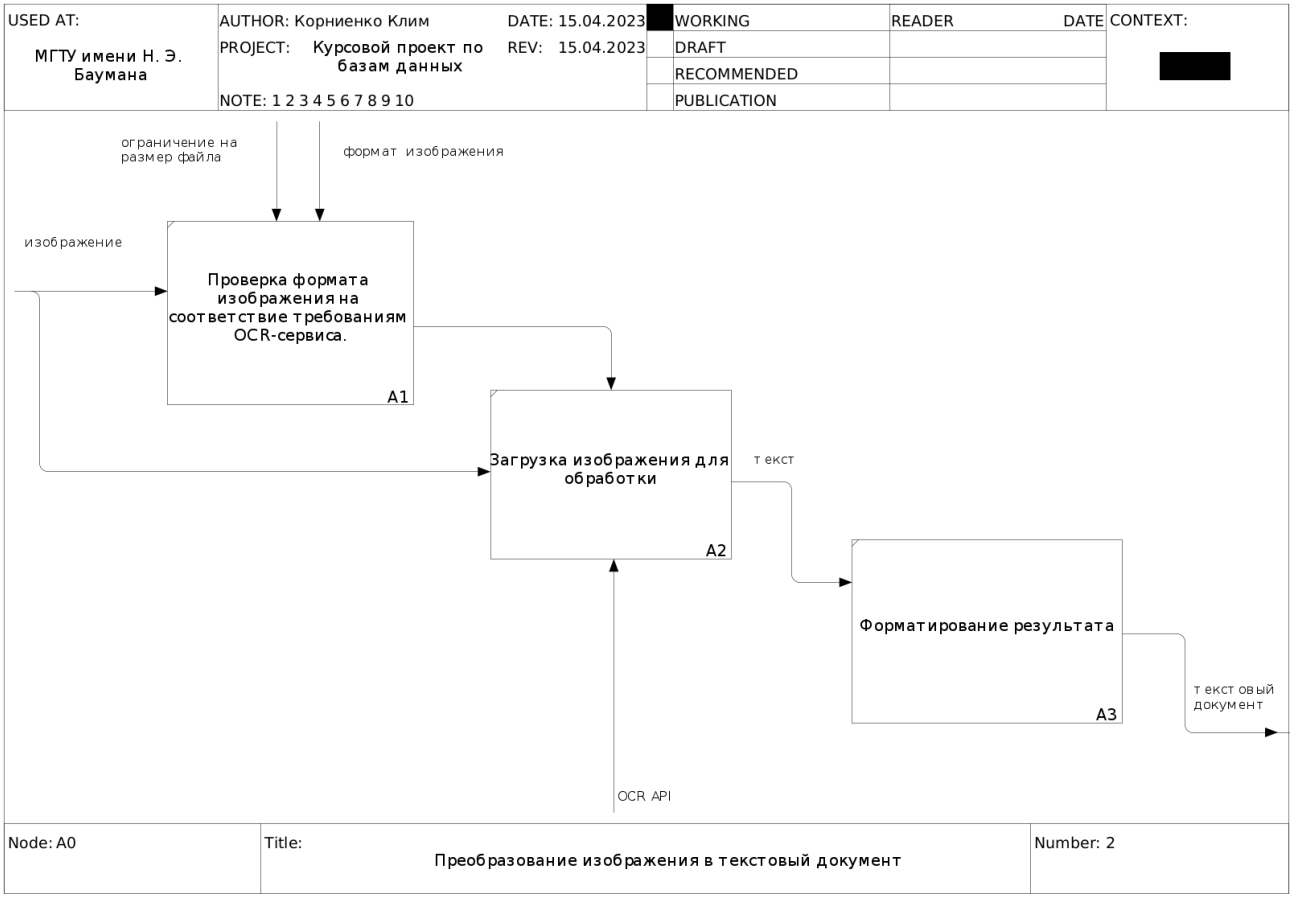

Рисунок 2.2 – Модель преобразования изображения в текст (первый уровень)

2.2 Диаграмма прецедентов

На рисунках 2.3 представлена диаграмма прецедентов для актора «пользователь».

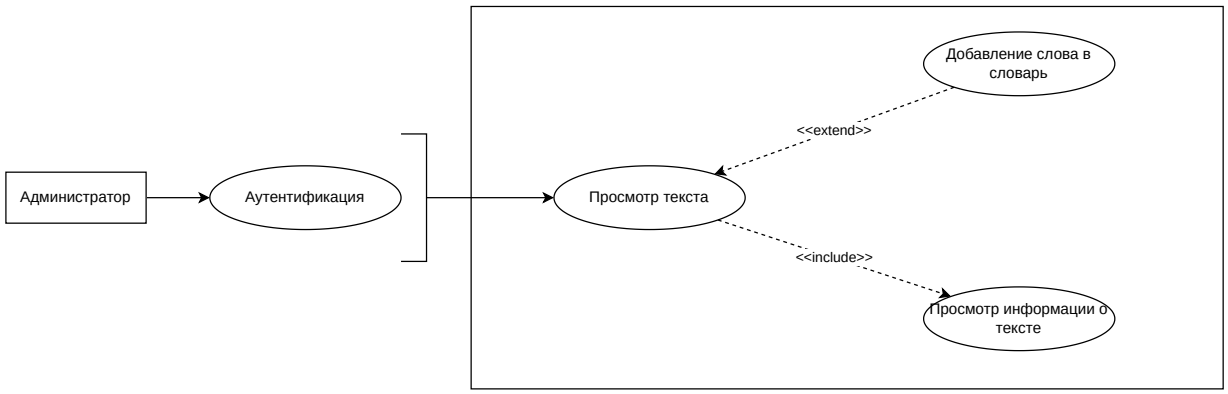

Рисунок 2.3 – Диаграмма прецедентов (пользователь)

На рисунке 2.4 представлена диаграмма прецедентов для актора «модератор».

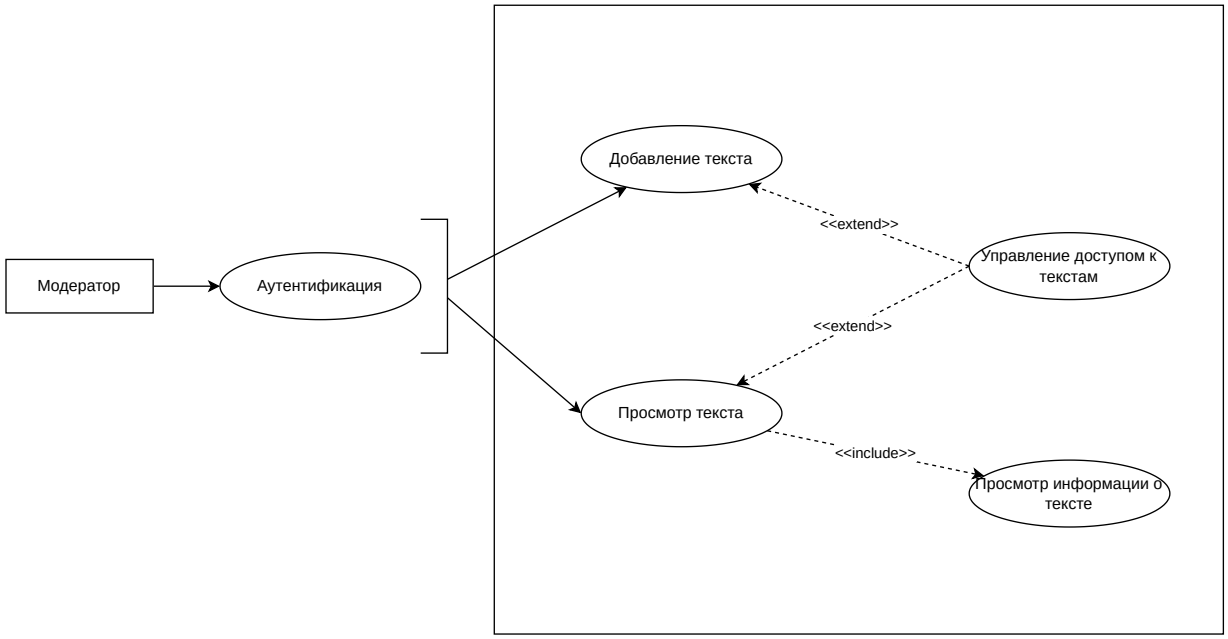

Рисунок 2.4 – Диаграмма прецедентов (модератор)

На рисунке 2.5 представлена диаграмма прецедентов для актора «администратор».

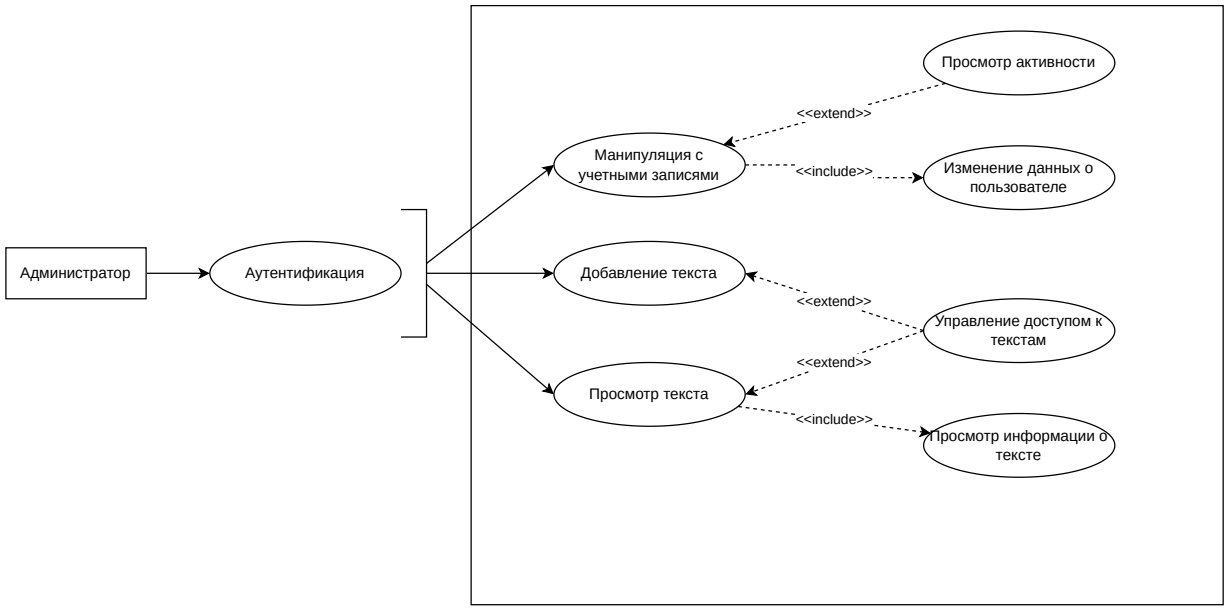

Рисунок 2.5 – Диаграмма прецедентов (администратор)

2.3 Схема базы данных

База данных содержит 7 таблиц, 2 из которые развязочные. Описание основных таблиц представлено ниже.

1. Таблица «KanjisDAO»:
   - содержит информацию о японских иероглифах;
   - служит основой для изучения иероглифов и обеспечивает доступ к информации о них.
2. Таблица «WordsDAO»:
   - содержит информацию о словах, состоящих из японских азбук и иероглифов;
   - данная таблица предоставляет доступ к различным словам, их переводам и чтениям.
3. Таблица «DictionariesDAO»:
   - содержит информацию о словах, добавленных в личный словарь конкретным пользователем;
   - таблица предоставляет доступ к сохраненным словам, для их последующего повторения.
4. Таблица «TextsDAO»:
   - содержит информацию о текстах;
   - таблица позволяет получать информацию о тексте: его название и содержание.
5. Таблица «UsersDAO»:
   - содержит информацию о пользователях;
   - таблица позволяет получать информацию о пользователе: его роль, логин, пароль.

На рисунке 2.6 представлена спецификация таблиц базы данных.

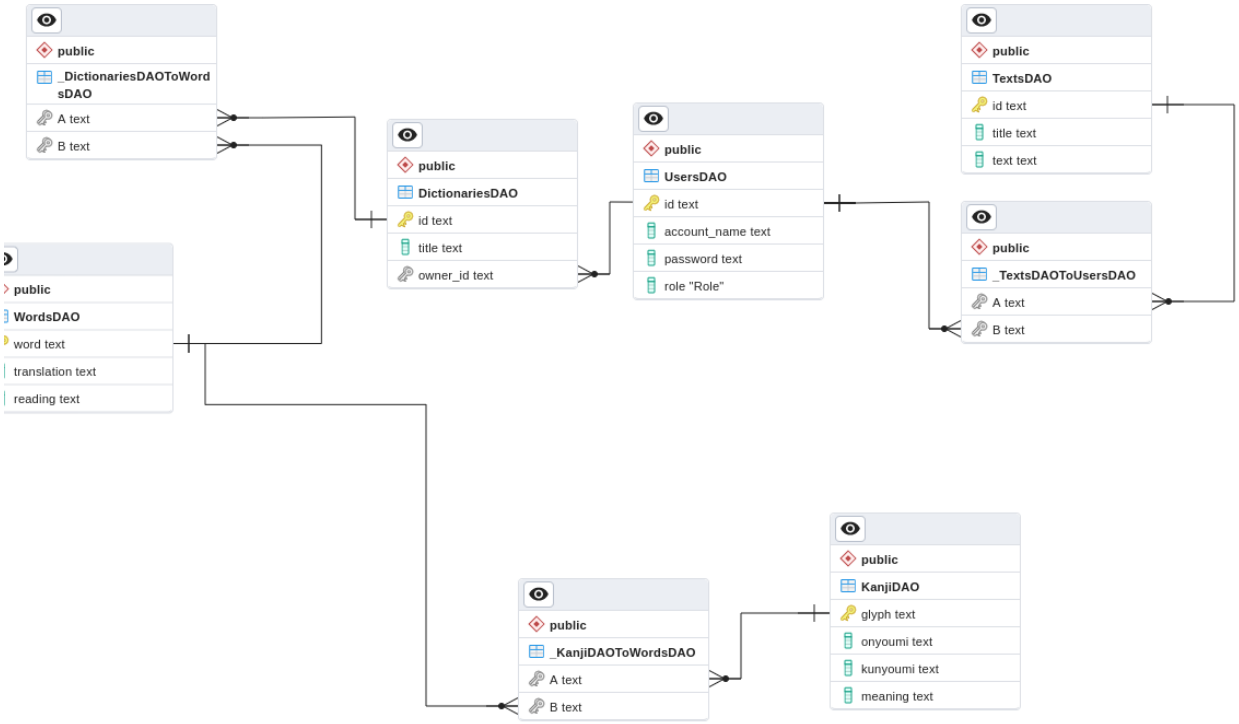

Рисунок 2.6 – Спецификация таблиц базы данных

2.4 Ограничения целостности

На описанные таблицы будут наложены следующие ограничения целостности:

1. В таблице «KanjisDAO» накладывается ограничение на уникальность поля `glyph`.

2. В таблице «DictionariesDAO» накладываются ограничения на уникальность полей `id`, `owner_id`. Также поле `owner_id` является внешним ключом для таблицы «UsersDAO».

3. В таблице «UsersDAO» накладываются ограничения на уникальность полей `id`, `account_name`.

4. В таблице «WordsDAO» накладывается ограничение на уникальность поля `word`.

5. В таблице «TextsDAO» накладывается ограничение на уникальность поля `id`.

2.5 Требования к ПО

К разрабатываемому ПО предъявляются следующие требования:

— приложение должно быть кроссплатформенным, поддерживать операционные системы семейства GNU/Linux, Android, а также иметь веб версию;

— для использования приложения необходимо пройти аутентификацию;

— пользователь должен иметь возможность получать информацию о словах в тексте нажатием на интересующее слово;

— при удалении текста модератором, все слова в словарях пользователей, относящиеся только к удаляемому тексту, также должны быть удалены.

Для удаления всех слов, связанных исключительно с удаляемым текстом, из словарей пользователей, будет использована хранимая функция и триггер в базе данных. При удалении текста модератором, триггер автоматически запускает хранимую функцию, которая удаляет все связанные слова из словарей пользователей. Это позволяет поддерживать актуальность словарей и исключать слова, которые больше не ассоциируются с каким-либо текстом.

Вывод из конструкторского раздела

В данном разделе была определена структура базы данных с таблицами для хранения информации о пользователях, иероглифах, словах, текстах и словарях, а также связи между ними. Были определены роли пользователя, модератора, администратора и их разрешения. Для оптического преобразования изображения будет использован внешний сервис.

3 Технологический раздел

3.1 Выбор средств реализации

В качестве языка разработки был выбран Dart в сочетании с фреймворком Flutter для обеспечения кроссплатформенности приложения. Использования таких средств разработки имеет ряд преимуществ:

— одновременная разработка для различных платформ. Flutter является кроссплатформенным фреймворком, что позволяет быстро разрабатывать приложения для множества платформ с использованием единой кодовой базы;

— быстрая разработка интерфейса пользователя. Flutter дает возможность декларативного описания элементов интерфейса, а также располагает набором готовых графических компонентов, которые легко настраивать и комбинировать для создания интерфейса пользователя;

— Dart является компилируемым языком, что позволяет достичь высокого уровня производительности;

— Dart позволяет интегрировать сторонние библиотеки и сервисы, которые могут быть полезными при разработке.

Выбор Dart и Flutter для разработки мобильного приложения для изучения японского языка обусловлен кроссплатформенностью, быстрой разработкой интерфейса, производительностью и расширяемостью. Эти факторы помогут ускорить разработку, обеспечить хороший пользовательский опыт и упростить поддержку приложения на разных платформах.

3.2 База данных

Для доступа к базе данных со стороны приложения была использована технология объектно-реляционного отображения, предоставляемого библиотекой Dart-orm и PrismaORM. На листинге 3.1 представлено описание схемы базы данных.

Листинг 3.1 – Исходный код описания схемы базы данных

```
generator client {
  provider = "dart run orm"
  output   = "../lib/data/repositories/prisma"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

enum Role {
  USER
  MODERATOR
  ADMIN
}

model UsersDAO {
  id           String           @id @default(uuid())
  account_name String           @unique
  password     String
  role         Role
  dictionary   DictionariesDAO?
  texts        TextsDAO[]
}

model KanjiDAO {
  glyph    String @id
  onyoumi  String
  kunyoumi String
  meaning  String
  words    WordsDAO[]
}

model DictionariesDAO {
```

```
  id        String     @id @default(uuid())
  title     String
  owner_id  String     @unique
  owner     UsersDAO   @relation(fields: [owner_id], references:
     [id])
  words     WordsDAO[]
}

model WordsDAO {
  word        String            @id
  translation String
  reading     String
  dicts       DictionariesDAO[]
  kanjis      KanjiDAO[]
}

model TextsDAO {
  id      String     @id @default(uuid())
  title   String
  text    String
  readers UsersDAO[]
}
```

3.2.1 Связь с приложением

Со стороны приложения был использован шаблон проектирования «репозиторий», предоставляющий интерфейс доступа к базе данных. Таким образом приложение не зависит от выбора конкретного способа взаимодействия с базой данных, описывая нужные для работы приложения методы, конкретная реализация которых может быть заменена. На листингах 3.2–3.5 представлен интерфейс доступа некоторым из таблиц базы данных.

Листинг 3.2 – Интерфейс доступа к таблице пользователей

```
import 'package:kango/data/entities/user.dart';

abstract class UserRepository {
  Future<User> register(User u);
  Future<User> findByAccountName(String accountName);
  Future<List<User>> findAll();
  Future<List<User>> findRegularUsers();
  Future<void> updateUser(String oldLogin, User newUser);
```

```
  Future<void> deleteUser(String login);
}
```

Листинг 3.3 – Интерфейс доступа к таблице слов

```
import 'package:kango/data/entities/word.dart';

abstract class WordRepository {
  Future<List<Word>> findAll();
  Future<Word> findByWord(String word);
  Future<void> insertWord(Word word);
  Future<void> updateMeaning(Word word, String newMeaning);
}
```

Листинг 3.4 – Интерфейс доступа к таблице текстов

```
import 'package:kango/data/entities/text.dart';
import 'package:kango/data/entities/user.dart';

abstract class TextRepository {
  Future<List<Text>> findAll();
  Future<List<Text>> findAccessibleByUser(String userLogin);
  Future<List<User>> findAccecients(String textId);
  Future<Text> findByID(String id);
  Future<String> insertText(Text text);
  Future<void> deleteById(String id);
  Future<void> updateText(String id, String newTitle, String
     newContent);
  Future<void> updateAccecients(String textId, List<User>
     newReaders);
}
```

Листинг 3.5 – Интерфейс доступа к таблице слов

```
import 'package:kango/data/entities/word.dart';

abstract class WordRepository {
  Future<List<Word>> findAll();
  Future<Word> findByWord(String word);
  Future<void> insertWord(Word word);
  Future<void> updateMeaning(Word word, String newMeaning);
}
```

3.2.2 Триггер

Для удаления всех слов, ассоциированных только с удаляемым текстом, из словарей пользователей, была написана хранимая функция и триггер, представленные на листинге 3.6.

Листинг 3.6 – Описание хранимой функции и триггера

```
create or replace function delete_words_in_text()
returns trigger as $$
begin
    -- Get all words that belong only to the specified text
       record
    create temp table words_to_delete on commit drop as
    select *
    from "WordsDAO" w
    where OLD.text like '%' || w.word || '%'
        and not exists(
            select *
            from "TextsDAO" t
            where t.text like '%' || w.word || '%'
                and t.id <> OLD.id
        );

    -- Delete all connections with dictionaries from database
    delete from "_DictionariesDAOToWordsDAO" as d
    where d."B" in (select word from words_to_delete);

    -- Delete all words that belong only to the specified text
       record
    delete from "WordsDAO" as w
    where w.word in (select word from words_to_delete);
    return OLD;
end;
$$ language plpgsql;


-- create trigger
create trigger delete_words_in_text_trigger
before delete on "TextsDAO"
for each row
execute function delete_words_in_text();
```

3.3 Аутентификация

В рамках данной работы был написан сервис, взаимодействующий с таблицей пользователя в базе данных. Сервис, представленный на листинге 3.7 поддерживает ролевую модель со стороны приложения.

Листинг 3.7 – Сервис аутентификации

```
import 'package:kango/data/entities/user.dart';
import 'package:kango/data/repositories/user.dart';

abstract class IAuthService {
  Future<User?> login(String username, String password);
  void logout();

  User get currentUser;

  bool get isModerator;
  bool get isAdmin;
}

class AuthService extends IAuthService {
  final UserRepository _userRepository;

  User? _currentUser;

  AuthService(this._userRepository);

  @override
  Future<User?> login(String username, String password) async {
    try {
      final user = await
         _userRepository.findByAccountName(username);
      if (user.password == password) {
        _currentUser = user;
        return user;
      }
      return null;
    } catch (e) {
      return null;
    }
  }
```

```
  @override
  void logout() {
    _currentUser = null;
  }

  @override
  User get currentUser {
    if (_currentUser == null) {
      throw Exception('user is not logged in');
    } else {
      return _currentUser!;
    }
  }

  @override
  bool get isModerator {
    return currentUser.role == UserRole.moderator;
  }

  @override
  bool get isAdmin {
    return currentUser.role == UserRole.admin;
  }
}
```

3.4 Тестирование

Для тестирования функционала приложения были написаны интеграционные тесты, проверяющие взаимодействие реализаций репозиториев API, предоставляющим функционал распознавания иероглифов, разбиение на отдельные слова и поиск значений в словаре Jisho. На листингах 3.8–3.10 представлены описанные выше интеграционные тесты.

Листинг 3.8 – Интеграционные тесты (часть 1)

```
import 'package:flutter_dotenv/flutter_dotenv.dart';
import 'package:test/test.dart';

import 'package:kango/vendor/goo/goo.dart';

void main() async {
  await dotenv.load(fileName: '.env');

  test('Goo API works', () async {
    const text = '';
    const expectedWordList = [
      WordDescription(word: "", reading: ""),
      WordDescription(word: "", reading: ""),
      WordDescription(word: "", reading: ""),
      WordDescription(word: "", reading: ""),
      WordDescription(word: "", reading: ""),
      WordDescription(word: "", reading: ""),
      WordDescription(word: "", reading: ""),
    ];

    final actual = await Goo.splitText(text);

    expect(actual, expectedWordList);
  });
}
```

Листинг 3.9 – Интеграционные тесты (часть 2)

```
import 'package:flutter_dotenv/flutter_dotenv.dart';
import 'package:kango/vendor/jisho/jisho.dart';
import 'package:test/test.dart';

void main() async {
  await dotenv.load(fileName: '.env');

  test('Jisho API works', () async {
    const word = "";
    const expectedDefinitions = [
      "commodity",
      "article of commerce",
      "goods",
      "stock",
      "merchandise"
    ];

    final actualDefinitions = await
       Jisho.getWordDefinitions(word);
    expect(actualDefinitions, expectedDefinitions);
  });
}
```

Листинг 3.10 – Интеграционные тесты (часть 3)

```
import 'dart:io';

import 'package:flutter/services.dart';
import 'package:flutter_dotenv/flutter_dotenv.dart';
import 'package:kango/vendor/ocr/ocr_space.dart';
import 'package:test/test.dart';

Future<File> loadAssetFile(String path) async {
  final data = await rootBundle.load(path);
  final buffer = data.buffer;
  return File('/tmp/flutter_ocr/$path')
    ..createSync(recursive: true)
    ..writeAsBytesSync(
        buffer.asUint8List(data.offsetInBytes,
           data.lengthInBytes));
}
```

```
void main() async {
  await dotenv.load(fileName: '.env');

  test('OCR API works', () async {
    File file = await loadAssetFile('lib/assets/test/img.png');

    final result = await OCRSpace().analyzeImage(file);

    const expected =
        "\r\n\r\n\r\n";

    expect(result.ok, true);
    expect(result.parsedText, expected);
    expect(result.error, "");
  });
}
```

Все интеграционные тесты были пройдены успешно.

Вывод из технологического раздела

В данном разделе был выбран язык Dart в сочетании с фреймворком Flutter для реализации логики приложения и интерфейса пользователя соответственно. Также был описан интерфейс доступа к базе данных и произведено интеграционное тестирование, отражающее правильное взаимодействие разрабатываемого программного обеспечения с внешними сервисами для оптического распознавания иероглифов, разбиение на отдельные слова и поиск значений в словаре японских слов.

4 Исследовательский раздел

4.1 Постановка исследования

Исследуется среднее время работы различных этапов процесса добавления текста в мобильное приложение из файла формата png:

— оптическое распознавание иероглифов из изображение и преобразование их в текст;

— вставка полученного в результате оптического распознавания текста в базу данных;

— разбиение текста на отдельные слова;

— вставка слов, выделенных из текста, в базу данных.

Для проведения замеров использованы пять изображений из файлов формата png, примерно одинакового качества. Результатом исследования является среднее значение времени работы каждого из этапов работы разработанного программного обеспечения, по полученным результатам построена гистограмма.

4.2 Результаты исследования

В ходе проведения исследования был написан тестовый модуль, представленный на листинге 4.1.

Листинг 4.1 – Модуль для исследования характеристик ПО

```
import 'dart:io';

import 'package:flutter_dotenv/flutter_dotenv.dart';
import
   'package:kango/data/repositories/prisma/prisma_client.dart';
import 'package:kango/data/repositories/prisma/text.dart';
import 'package:kango/data/repositories/prisma/word.dart';

import '../vendor/ocr/ocr_space_test.dart';
import 'wrappers/service/text_provider.dart';

void main() async {
  await dotenv.load(fileName: '.env');

  final prisma = PrismaClient(
    datasources: Datasources(
      db: dotenv.env['DATABASE_URL'],
    ),
  );

  // 1 инициализация репозиториев
  final textRepo = PrismaTextRepository(prisma: prisma);
  final wordRepo = PrismaWordRepository(prisma: prisma);

  final textProvider = WrappedTextsProvider(textRepo, wordRepo);

  await prisma.textsDAO.deleteMany(
    where: const TextsDAOWhereInput(
      title: StringFilter(startsWith: '__research_text_'),
    ),
  );

  final resultList = <TimestampTuple>[];
  for (var i in [1, 2, 3, 4, 5]) {
    final imageFilename = 'lib/assets/research/img_$i.png';
    File file = await loadAssetFile(imageFilename);
```

```
    final result =
        await
           textProvider.uploadTextFromFile('__research_text_$i',
           file);

    resultList.add(result);
    print('done $i');
  }

  await prisma.textsDAO.deleteMany(
    where: const TextsDAOWhereInput(
      title: StringFilter(startsWith: '__research_text_'),
    ),
  );

  serializeTimeTuple(resultList);
}

void serializeTimeTuple(List<TimestampTuple> tuples) {
  final file = File('test/research/results.csv');
  final sink = file.openWrite();

  sink.writeln('start,ocrPerformed,textInserted,wordSplit,
    wordTranslated,end');

  for (final tuple in tuples) {
    sink.writeln(
        '${tuple.start.microsecondsSinceEpoch},${tuple
          .ocrPerformed.microsecondsSinceEpoch},${tuple
          .textInserted.microsecondsSinceEpoch},${tuple
          .wordSplit.microsecondsSinceEpoch},${tuple
          .end.microsecondsSinceEpoch}');
  }

  sink.close();
}
```

Гистограмма, отображающая результат исследования, показана на рисунке 4.1

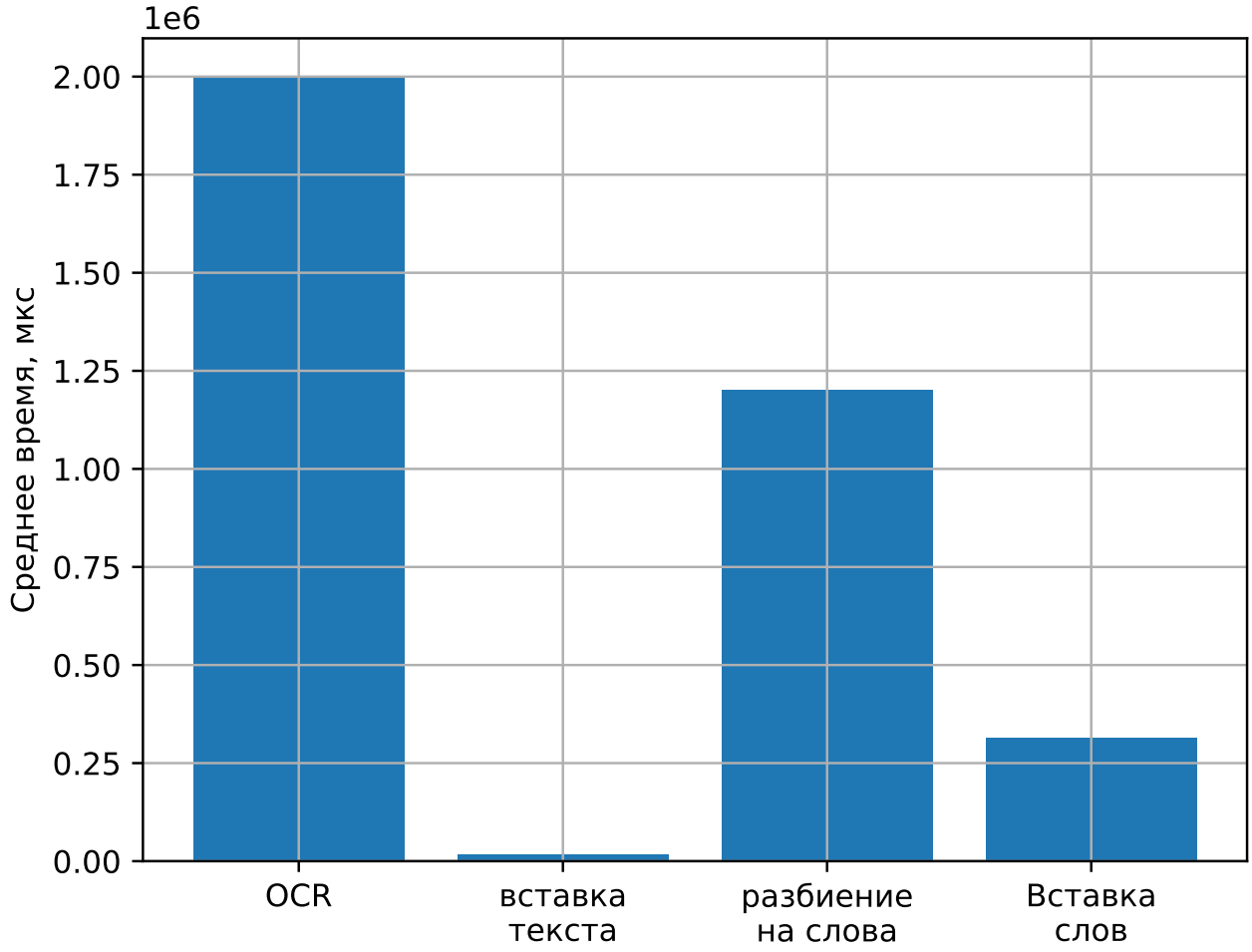

Рисунок 4.1 – Гистограмма исследования характеристик ПО

Исходя из гистограммы, изображенной на рисунке 4.1, можно заявить, что оптическое распознавание текста занимает значительную часть времени обработки текста в приложении. Кэширование может существенно уменьшить время данного этапа работы программного обеспечения путем сохранения результатов предыдущих распознаваний и их использовании при обработке схожих или идентичных изображений. Кэширование может быть проведено как на уровне приложения, так и на уровне промежуточного слоя, хранящего результаты предыдущих распознаваний.

На рисунке 4.2 представлена зависимость времени вставки текста в базу данных от количества иероглифов в тексте.

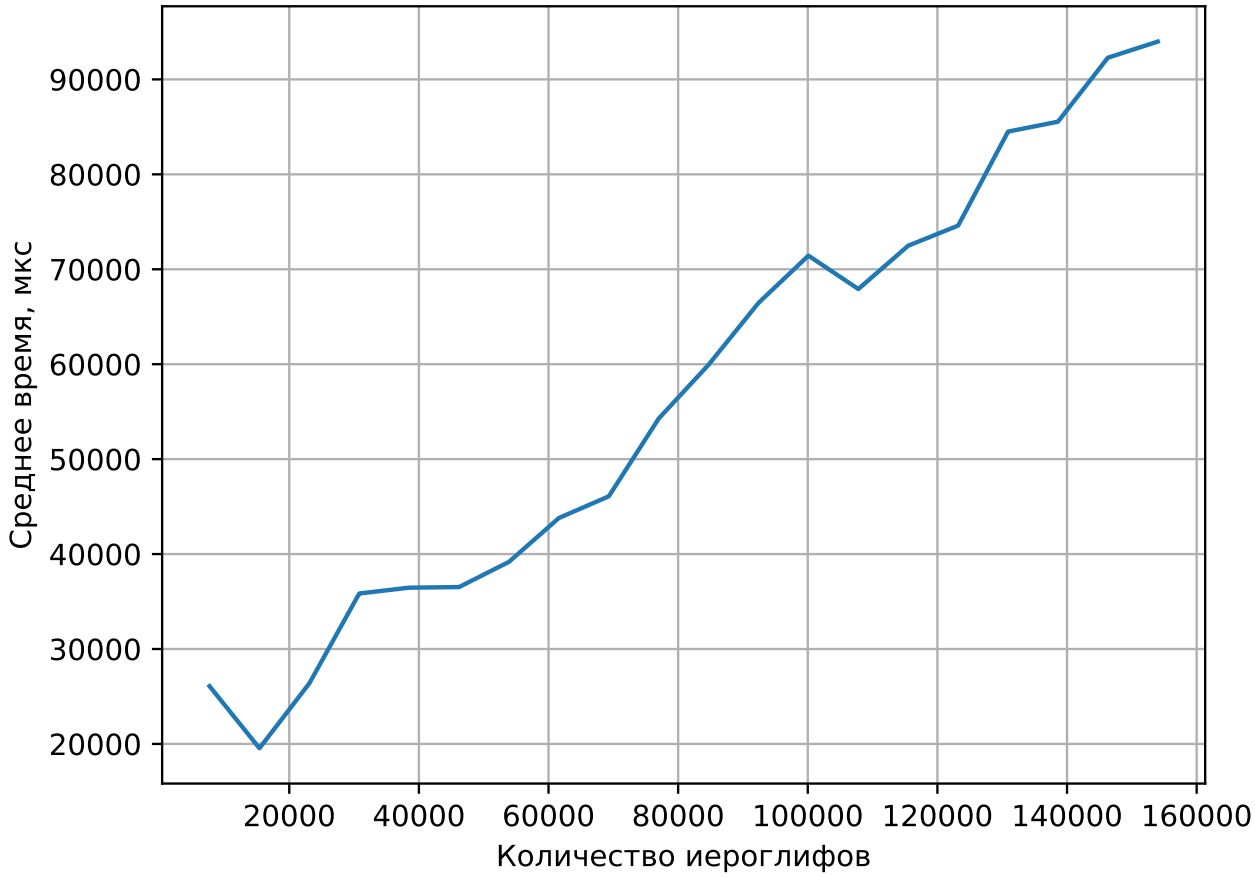

Рисунок 4.2 – График зависимости времени вставки текста в базу данных от количества иероглифов в тексте

Вывод из исследовательского раздела

По полученным в ходе исследования данным было выяснено, что наибольшую часть времени работы программного обеспечения занимает оптическое распознавание иероглифов, поэтому стоит рассмотреть возможность кэширования уже распознанных иероглифов. Таким образом, при повторном обращении к уже известным изображениям, время распознавания будет сокращено. Данные исследования свидетельствуют о линейной зависимости времени вставки текста в базу данных от количества иероглифов в тексте.

ЗАКЛЮЧЕНИЕ

В ходе работы было разработано мобильное приложение для упрощения изучения японского языка. Были выполнены следующие задачи:

— было проведено исследование, чтобы получить глубокое понимание требований и особенностей изучения японского языка, что помогло определить ключевые функциональные возможности и основные компоненты приложения;

— спроектировано взаимодействие между различными компонентами приложения, включая пользовательский интерфейс, обработку данных и базу данных, также была разработана структура базы данных для хранения информации, необходимой для изучения японского языка;

— разработан интерфейс, который обеспечивает доступ к базе данных и позволяет пользователю взаимодействовать с содержимым приложения и проведено тестирование функциональности системы;

— изучена производительность приложения, что позволило выявить сильные и слабые стороны приложения, а также потенциальные области для улучшений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Мухтарова А. А.* Использование ИТ-технологий при изучении английского и японского языка // Умная цифровая экономика. — 2022. — Т. 2, № 3. — С. 34—41.
2. *Cheng T.* Applying networked learning to improve learner interactions: A new paradigm of teaching and learning in ODL // Asian Association of Open Universities Journal. — 2013. — Т. 8. — С. 67—85. — DOI: 10.1108/AAOUJ-08-02-2013-B006.
3. Recognizing modern Japanese magazines by combining Deep Learning with language models / N. T. Nguyen [и др.] // 2021 13th International Conference on Knowledge and Systems Engineering (KSE). — 2021. — С. 1—6. — DOI: 10.1109/KSE53942.2021.9648643.
4. *Sasanti N. S.* Japanese Language Learning Consistency in the Digital Era // Jurnal Ilmiah Lingua Idea. — 2022. — Т. 13. — С. 207—219.
5. *Febrianty F.*, *Ricardo R.* Information Technology for Japanese Learning // IOP Conference Series: Materials Science and Engineering. — 2019. — Т. 662, № 2. — С. 022117. — DOI: 10.1088/1757-899X/662/2/022117. — URL: https://dx.doi.org/10.1088/1757-899X/662/2/022117.
6. *Дейт К. Д.* Введение в системы баз данных. — М. : Издательский дом "Вильямс", 2005.
7. *Elmasri R.*, *Navathe S. B.* Fundamentals of Database Systems / под ред. M. Hirsch. — London : Pearson, 2010.
8. *A Malik A Burney F. A.* A Comparative Study of Unstructured Data with SQL and NO-SQL Database Management Systems // Journal of Computer and Communications. — 2020. — Т. 8. — С. 59—71. — DOI: 10.4236/jcc.2020.84005.
9. *Мейер М.* Теория реляционных баз данных. — М. : Мир, 1987.
10. *К Н Васильева Г. Я. Х.* Реляционные базы данных // Colloquium-Journal. — Warszawa, 2020. — С. 22—23.
11. Microsoft: SQL Server. [Электронный Ресурс]. — Режим Доступа: https://www.microsoft.com/en-us/sql-server/ (дата обращения: 13.04.2023).

12. MariaDB Server: The open source relational database. [Электронный Ресурс]. — Режим Доступа: https://mariadb.org/ (дата обращения: 13.04.2023).

13. PostgreSQL: The World's Most Advanced Open Source Relational Database. [Электронный Ресурс]. — Режим Доступа: https://www.postgresql.org/ (дата обращения: 13.04.2023).

14. Database | Oracle. [Электронный Ресурс]. — Режим Доступа: https://www.oracle.com/database/ (дата обращения: 16.04.2023).

15. IBM Db2 | IBM. [Электронный Ресурс]. — Режим Доступа: https://www.ibm.com/products/db2 (дата обращения: 16.04.2023).

16. SQLite Home Page. [Электронный Ресурс]. — Режим Доступа: https://sqlite.org/index.html (дата обращения: 16.04.2023).

17. *Ramya Gangaiamaran M. P.* Review on Use of Mobile Apps for Language Learning // International Journal of Applied Engineering Researc. — 2017. — Т. 12, № 21. — С. 11242—11251. — ISSN 0973-4562.

18. Duolingo - The world's best way to learn a language. [Электронный Ресурс]. — Режим Доступа: https://www.duolingo.com/ (дата обращения: 14.04.2023).

19. Learn a language. Memrise is authentic, useful & personalised. [Электронный Ресурс]. — Режим Доступа: https://www.memrise.com/ (дата обращения: 14.04.2023).

20. LingoDeer: Learn Japanese, Korean, Chinese and more. [Электронный Ресурс]. — Режим Доступа: https://www.lingodeer.com/ (дата обращения: 14.04.2023).

21. WaniKani, a kanji learning application by Tofugu. [Электронный Ресурс]. — Режим Доступа: https://www.wanikani.com/ (дата обращения: 14.04.2023).

22. *Ohta A. S.* Second Language Acquisition Processes In The Classroom: Learning Japanese. — London : Lawrence Erlbaum Associates, 2001.

ПРИЛОЖЕНИЕ А