**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА АНАЛИЗА ДАННЫХ И ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ

Направление: 09.03.03 – Прикладная информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**Экранная клавиатура для мобильных устройств с поддержкой лингвистических сервисов**

**Работа завершена:**

Студент 4 курса

группы 09-951

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фархетдинов Р.Р.

**Работа допущена к защите:**

Научный руководитель

ученая степень, ученое звание,

должность

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сафина Л.И.

Заведующий кафедрой

ученая степень, ученое звание

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Вахитов Г.З.

Казань – 2023

Содержание

[Введение 3](#_Toc132986598)

[1. Формирование требований 5](#_Toc132986599)

[1.1. Исследование предметной области лингвистических сервисов 5](#_Toc132986600)

[1.2. Анализ существующих клавиатур с поддержкой лингвистических сервисов 8](#_Toc132986601)

[1.3. Техническое задание 10](#_Toc132986602)

[2. Проектирование сервиса-клавиатуры 13](#_Toc132986603)

[2.1. Проектирование баз данных 13](#_Toc132986604)

[2.2. Проектирование диаграмм сервиса-клавиатуры 15](#_Toc132986605)

[2.3. Пользовательское взаимодействие с клавиатурой 20](#_Toc132986606)

[3. Реализация сервиса клавиатуры 25](#_Toc132986607)

[3.1. Модуль серверной части 25](#_Toc132986608)

[3.2. Модуль клиентской части 27](#_Toc132986609)

[3.3. Модуль взаимодействия 29](#_Toc132986610)

[4. Тестирование сервиса-клавиатуры 32](#_Toc132986611)

[4.1. Тестирование модуля серверной части 32](#_Toc132986612)

[4.2. Тестирование модуля клиентской части 32](#_Toc132986613)

[4.3. Тестирование модуля взаимодействия 34](#_Toc132986614)

[Заключение 34](#_Toc132986615)

[Список использованных источников 34](#_Toc132986616)

# Введение

История всех экранных клавиатур начинается с 1992 года — года создания первой оптической виртуальной клавиатуры, изобретенной и запатентованной инженерами IBM. Хотя данный вид клавиатуры появился относительно недавно, история её развития достаточно обширна.

Первые виртуальные клавиатуры не обладали возможностью поддерживать какие-либо лингвистические сервисы. Чуть позже появились клавиатуры с данной возможностью. Но с покупкой нового устройства (смартфона, компьютера) все данные о стиле печати пользователя оставались на старых устройствах. Отличным решением было бы сохранять данные пользователя в «облаке», чтобы избавиться от зависимости от определенного устройства. В эпоху Интернета синхронизация данных в сети является неотъемлемой частью большинства людей.

Целью выпускной квалификационной работы является разработка экранной клавиатуры для мобильных устройств с поддержкой лингвистических сервисов. Такая клавиатура нацелена помогать любым пользователям набирать текст на мобильных устройствах максимально продуктивно: быстро, без ошибок, с индивидуальным «почерком» пользователя. Разработка экранной клавиатуры актуальна, потому что с появлением социальных сетей, мессенджеров и сенсорных телефонов люди во всем мире стали активно переписываться друг с другом целыми днями, тем самым клавиатура является необходимым элементом в жизни любого современного человека. Профессиональная терминология, новые слова в разных сферах, имена, фамилии, клички, названия брендов – все эти наименования не могут быть обобщены и заданы общими правилами. Персонализированная клавиатура позволяет хранить личный «словарный запас» в облаке и пользоваться им с любого устройства.

Для разработки данной системы выделяют следующие задачи:

1. Изучение предметной области и анализ существующих решений в разработке экранной клавиатуры;
2. Составление технического задания к функциям и интерфейсу программы;
3. Проектирование системы «экранная клавиатура»;
4. Выбор программных средств реализации;
5. Написание программного кода;
6. Тестирование приложения и его отладка.

# 1. Формирование требований

## 1.1. Исследование предметной области лингвистических сервисов

В данной предметной области будут задействованы три основных лингвистических сервиса:

1. Орфокорректор — когда пользователь вводит слово, и оно неправильное, ему рекомендуются до трех правильных вариантов пока курсор находится в конце этого слова;
2. Предиктивный ввод — когда пользователь ввел правильное слово, ему рекомендуются три слова на выбор, каждое из которых может идти за этим словом;
3. Дополнение — когда пользователь набирает слово и, не дожидаясь его окончания, клавиатура начинает предлагать до трех вариантов слов на выбор, которые содержат в себе только что набираемое пользователем слово. Тем самым пользователю предоставляется возможность поскорей закончить слово, выбрав наиболее подходящее из списка.

Экранная клавиатура должна поддерживать вышеперечисленные лингвистические сервисы и «затачиваться» под определенного пользователя, который пользуется данной клавиатурой.

Разработка экранной клавиатуры в данной предметной области призвана решить проблему с набором текста на мобильных устройствах: медленный набор текста и утомительное исправление ошибок, а также индивидуальный «почерк» письма пользователя.

Постоянный набор слов требует определенных усилий со стороны пользователя, который, в свою очередь, хочет вводить слова быстро и корректно; в некоторых случаях предложения исправления слова как «неправильного», относящегося к сленгу пользователя, неуместно – всё это определяет практическую значимость данной клавиатуры.

Также можно затронуть одну интересную статью [-], где два студента-математика, учащиеся в Кембриджском университете, Фолькер Шлуе и Майкл Фестер разработали новый способ набора текста в мобильные устройства, а именно — ПО 8-pen, являющееся виртуальной клавиатурой. ПО 8pen имеет достаточно сильные расхождения от обычных клавиатур в плане организации кнопок — выглядит как «икс», вдоль линий которого расположены буквы от центра «икса» до края по убыванию частоты встречаемости, а в центре «икса» расположена большая точка черного цвета. По углам расположены такие кнопки, как «shift», «delete», «enter» и «?123». Принцип работы следующий: для того чтобы ввести символ, нужно нажать на черную точку в центре «икса» и произвести переход в один из секторов (всего их 4). Число секторов определяется порядком букв на «иксе», а направление движения — стороной, где размещены эти буквы.

Разумеется, в начале работы с данной клавиатурой пользователю может быть не комфортно, так как она в разрез отличается от классических клавиатур. Со временем пользователь начнет привыкать к данному ПО и будет набирать текст в разы быстрее, чем раньше. Также данная клавиатура занимает намного меньше площадь экрана, чем обычная, что добавляет свободу движения. Тем самым придумана новая возможность вводить текст быстро и без ошибок, не прибегая к лингвистическим сервисам.

В следующей обширной статье [-] рассказывается о способах ввода с помощью экранной клавиатуры и обсуждается ее дизайн. Разработка настольных компьютеров, особенно веб-разработка, приводит к вялому интерактивному дизайну: здесь делаются предположения, которые часто оказываются ложными за пределами этой узкой области. В эпоху массового распространения мобильных устройств такие предположения создают множество проблем. Одним из наиболее важных аспектов являются способы ввода информации.

iPhone по умолчанию использует виртуальную сенсорную панель для ввода информации. Этот взгляд очень интересен благодаря своей гибкости. Параметры ввода для определенных полей могут влиять на раскладку клавиатуры в целом и на функции отдельных кнопок в частности. Экранные клавиатуры могут использовать интересные способы взаимодействия с пользователем, начиная от набора текста непрерывным движением пальцев и заканчивая виртуальным колесом прокрутки, позволяющим выбирать значение, например, дату и время, из ограниченного набора элементов.

Клавиатуры используются для ввода слов, а панели используются для ввода цифр и символов. Хотя границы между различными функциями виртуальных клавиатур могут быть размыты, определение разницы между ними является довольно важной задачей.

Интересным моментом является изменение режима ввода. При переключении в новый режим могут измениться как названия, так и функции клавиш, а также расположение и размер кнопок. Это означает, что на самом деле количество режимов ввода бесконечно. Например, при вводе адреса электронной почты может отображаться клавиатура, включающая символ «@» и отдельную клавишу «.com», при этом в адресе могут отсутствовать недопустимые символы, например, запятая, в противном случае раскладка останется прежней.

Цифровой ввод — один из самых интересных режимов. Например, расположение кнопок для набора телефонного номера отличается от раскладки цифровой панели на клавиатуре. Иногда цифры расположены в верхнем ряду буквенно-цифровой панели, как на стандартной компьютерной клавиатуре.

Ввод жестов с виртуальной клавиатуры позволяет ускорить ввод текста (приложение Swype от Nuance studio). Пользователь вводит слово одним движением пальца, которое может остановиться или изменить направление на любом знаке. Дополнительный режим используется наряду со стандартным: пользователь всегда может вернуться к набору текста, нажимая клавиши поочередно — внешний вид клавиатуры не изменится.

Инструменты выбора даты и времени и другие механизмы ввода определенного значения позволяют пользователю произвольно выбирать способ взаимодействия, каждый из которых имеет свои особенности.

Некоторые из методов ввода, такие как колесо прокрутки даты и времени в Android, дополняют сенсорный язык и существуют наряду с ручным вводом. Один щелчок по значению открывает виртуальную клавиатуру или цифровую панель.

В статье [-] речь пойдет о типах элементов ввода с помощью экранной клавиатуры. Рассмотрим самые распространенные типы.

Date. Ни для кого не секрет, что календари на мобильных устройствах ужасно маленькие, что доставляет дискомфорт. Также есть требования на многих текстовых полях, которые принуждают пользователей при вводе даты соблюдать строгие правила проверки. Установив тип ввода date, мобильный браузер решит оставшиеся проблемы.

Email. Ввод адреса электронной почты на обычной клавиатуре требует усилий со стороны пользователя. Данный тип ввода добавляет необходимые клавиши на первый план, такие как «@» и «.com», тем самым ввод адреса значительно упрощается.

Text. Это, пожалуй, самый часто встречающийся тип элементов ввода, который задействован везде.

Tel. Если нужно получить от пользователя только число, то нужно использовать тип элемента ввода tel, так как данный тип предоставляет пользователю цифровую клавиатуру.

## 1.2. Анализ существующих клавиатур с поддержкой лингвистических сервисов

В статье [-] рассмотрена одна из клавиатур с поддержкой лингвистических сервисов — Microsoft SwiftKey. Microsoft SwiftKey — это интеллектуальная клавиатура, изучающая стиль написания текста определенного пользователя, тем самым помогая ему быстрее и продуктивнее печатать текст. С помощью этой персонализированной клавиатуры есть возможность печатать сообщения, отправлять GIF-файлы и Emoji и так далее. Microsoft SwiftKey постоянно анализирует и подстраивается под уникальный стиль набора своего пользователя, сохраняя в памяти используемый им сленг и Emoji. Данная клавиатура предоставляет расширение сервиса орфокорректор в виде автоисправления. Настраиваемая клавиатура выдает полезные подсказки, помогающие печатать текст быстро и без ошибок. Основные функции этой клавиатуры:

1. Клавиатура Emoji подстраивается к стилю печати пользователя, обучается и предоставляет его любимые эмотиконы во время общения;
2. Проверка орфографии и автоматическая подстановка текста с прогнозированием на основе искусственного интеллекта;
3. Имеет поддержку около 400 языков;
4. Около 100 ярких тем.

В сравнение с Microsoft SwiftKey возьмем обычную встроенную экранную клавиатуру на более-менее современные Android-смартфоны от компании Google под названием Gboard. Сразу можно сказать, что Gboard также поддерживает все три лингвистических сервиса (орфокорректор, предиктивный ввод и дополнение), как и вышеописанная клавиатура.

Данная стандартная клавиатура также имеет возможность обучаться под определенного пользователя. Нужно просто активировать пункт Персонализация в ее настройках. Gboard будет некоторое время хранить на устройстве пользователя фрагменты напечатанного и продиктованного им текста.

Рассмотрим еще одну клавиатуру — экранная клавиатура Windows 10. Этот вид клавиатур поддерживает сервисы дополнение и предиктивный ввод, но не обучаем. Также есть возможность для более детальной ее настройки, например, использовать звуковой сигнал, если пользователь хочет слышать звук при нажатии клавиш.

Исходя из всего вышесказанного, можно сделать вывод, что на данный момент субъективно лучшей клавиатурой является Microsoft SwiftKey, так как она обладает наибольшим функционалом по сравнению со стандартными экранными клавиатурами на смартфонах и компьютерах. Наиболее отличительная особенность Microsoft SwiftKey — это показ статистики персонализированного ввода пользователю в наиболее удобном и развернутом виде, что отсутствует в других рассмотренных клавиатурах. Некоторые особенности всех трех вышеописанных клавиатур будут позаимствованы при выполнении выпускной квалификационной работы.

## 1.3. Техническое задание

Целью разработки экранной клавиатуры для мобильных устройств с поддержкой лингвистических сервисов является оптимизация, а также персонализация стиля ввода текста пользователем. Работа предлагает решение проблем с набором текста на мобильных устройствах: медленный набор текста и утомительное исправление ошибок. Функциональные требования к проектируемой системе:

1. Возможность набирать текст с помощью клавиш;
2. Рекомендации по исправлению слова до трех правильных вариантов;
3. Рекомендации до трех слов на выбор, каждое из которых может идти за текущим словом;
4. Рекомендации до трех слов на выбор, каждым из которых можно завершить написание текущего слова;
5. Обучение под конкретного пользователя, анализируя его набор текста и его использование рекомендаций, предложенных клавиатурой;
6. Возможность менять в настройках приложения параметры подсказок —фон, цвет текста, шрифт;
7. Должна иметься возможность авторизации и регистрации в приложении.

Данные о словах и необходимая информация о пользователях должны храниться в базах данных, причем пароли пользователей в целях безопасности должны храниться в виде хэша. Вся информация, используемая приложением, должна храниться в структурированном виде.

Далее обозначены требования к компонентам аппаратного и программного обеспечения, на которых развертывается программная реализация экранной клавиатуры:

1. ОС — Android;
2. Оперативная память — 2 GB;
3. Количество ядер — 2;
4. Тактовая частота процессора — 1000 МГц.

Данные от клиента на сервер, а с сервера в базу данных, и обратно должны передаваться в формате JSON. Протокол передачи данных, использующийся в приложении — http. Если возникнет сбой с сервером или с базами данных (на стороне сервера или локально), пользователь все равно сможет продолжить пользоваться клавиатурой, а также с подбором слов в качестве подсказок, кроме случая, если сбой произошел с локальной базой данных перед началом работы пользователя. Следует заметить, что обучаемость клавиатуры под пользователя при любых сбоях с локальной базой данных не предусмотрена.

Серверная часть (REST API) приложения будет разработана на языке программирования Java с использованием фрэймворка Spring.

Для взаимодействия с сервером выбрана СУБД PostgreSQL.

Клиентская часть будет также разработана на языке программирования Java и на локальном уровне будет обмениваться данными с СУБД SQLite. Чтобы взаимодействовать с сервером посредством отправки к нему запросов, а также для преобразования данных в формат JSON и обратно, клиентская часть будет использовать REST-клиент для Android и Java под названием Retrofit.

Интерфейс приложения должен быть удобен для конечного пользователя, который отвечает следующим требованиям:

1. Наличие удобного и не вызывающего напряженность глаз шрифта;
2. Использование русскоязычного интерфейса для авторизации и регистрации пользователя, а также для настройки клавиатуры;
3. Использование цветовой палитры спокойных оттенков для комфортного взаимодействия пользователя с программой.

# 2. Проектирование сервиса-клавиатуры

## 2.1. Проектирование баз данных

В приложении присутствуют две базы данных: на стороне сервера (серверная) и на стороне клиента (локальная). Разбирать диаграммы баз данных требуется для более глубокого понимания работы лингвистических сервисов. Сначала рассмотрим диаграмму серверной базы данных (рисунок 1).

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 1. Диаграмма серверной базы данных

Серверная база данных задействуется в случае, когда требуется выполнить авторизацию, регистрацию, синхронизацию данных или удаление аккаунта. Есть три таблицы: users, words, collocations.

Таблица users хранит логины и пароли пользователей, чтобы в дальнейшем их идентифицировать (необходимо для синхронизации данных определенного пользователя).

Таблица words хранит идентификаторы пользователей (userId), которым соответствуют определенные слова (word) и счетчик частоты появления для каждого слова (count). Данная таблица нужна для определения орфографически верного слова, а также для более быстрого завершения текущего слова. Благодаря этой сущности слово может определиться как «правильное» и «неправильное» для определенного пользователя. «Правильным» становится то слово, чей счетчик count больше или равен числу от 1 до 10 (пользователь может сам определить наиболее подходящее для себя, но по умолчанию число равно 3).

Таблица collocations хранит идентификатор на предыдущее слово (prevId), идентификатор на следующее слово (nextId) и счетчик данных пар (count). С помощью этой таблицы подбираются слова для продолжения, основываясь на частоту их использования (значение count) с определенным словом. В данной сущности могут находиться только «правильные» слова.

Теперь рассмотрим диаграмму локальной базы данных (рисунок 2).

Изображение выглядит как диаграмма, стол

Автоматически созданное описание

Рисунок 2. Диаграмма локальной базы данных

Локальная база данных взаимодействует с пользователем на постоянной основе, то есть всегда, независимо от того, вошел он в систему или нет, включил использование подсказок или отключил и так далее. Есть 4 таблицы: user, words, collocations и ime\_settings.

Таблица user, в целях большей безопасности, хранит только логин пользователя (без пароля), чтобы в дальнейшем его идентифицировать и иметь возможность синхронизировать данные.

Таблица words идентична своему аналогу в «облаке», за исключением, разве что, идентификатора пользователя userId, так как клавиатура будет обучаться независимо от того, зарегистрирован пользователь или нет.

Таблица collocations полностью идентична своему аналогу в «облаке».

Таблица ime\_settings нужна для хранения следующих данных:

1. Использование звука;
2. Использование вибрации;
3. Использование подсказок;
4. Определение цвета фона подсказок;
5. Определение цвета текста сервиса дополнение;
6. Определение цвета текста сервиса орфокорректора;
7. Определение цвета текста сервиса предиктивного ввода;
8. Определение шрифта текста для подсказок;
9. Задание скорости обучения.

## 2.2. Проектирование диаграмм сервиса-клавиатуры

В системе, как уже было сказано, работают три лингвистических сервиса: дополнение, орфокорректор и предиктивный ввод. Работа данных сервисов спроектирована на одной общей диаграмме деятельности. Для лучшего понимания функциональности каждого из этих сервисов диаграмма будет разбита на три части, и каждая часть будет разобрана по отдельности. Начнем с сервиса дополнение.

Дополнение предоставляет возможность ускорить написание слова (быстро закончить его), подобрав наиболее подходящие для этого рекомендации. Модель данного лингвистического сервиса изображена в виде диаграммы деятельности (рисунок 3).

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 3. Модель дополнения

Рассмотрим алгоритм. Пользователь открывает клавиатуру и может начать писать слово, либо выбрать одно из трех слов в качестве подсказок, представленных сервисом дополнение.

Подсказки подбираются следующим образом: из таблицы words (эта таблица никогда не может быть пустой, так как при установке приложения она инициализируется значениями из встроенного словаря) берутся три слова, начальная часть которых совпадает с тем, что последнее написал пользователь на текущий момент, а также чьи счетчики count являются максимальными по значению (частота использования слова). При выборе пользователем подсказки клавиатура подставит выбранное слово на место предыдущего и счетчик count выбранного слова увеличится на 1.

Но, независимо от выбора пользователя, происходит следующая проверка: если символ в конце строки не равен символу алфавита и длина последнего слова не равна 0, то запускается сервис орфокорректор, иначе работа цикла продолжается и подбираются все новые слова для дополнения. Далее рассмотрим сервис орфокорректор.

Орфокорректор позволяет предлагать слова для исправления текущего неправильно написанного слова. Также, если несколько раз не исправлять определенное слово, то орфокорректор «запомнит» его и не будет в дальнейшем предлагать его исправление — еще одна особенность данного лингвистического сервиса. Модель этого сервиса изображена в виде диаграммы деятельности (рисунок 4).

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 4. Модель орфокорректора

Рассмотрим алгоритм. Если слово определилось как «правильное», то счетчик count данного слова увеличивается на 1, орфокорректор заканчивает свою работу и следом запускается сервис предиктивный ввод. Если слово определилось как «неправильное», то у пользователя появляется два варианта действий: либо нажать на клавишу, либо выбрать слово из только что подобранных рекомендаций для исправления в порядке слева направо по убыванию вероятности наиболее подходящего слова.

Подсказки подбираются следующим образом: начинается поиск в таблице words последнего написанного пользователем слова. Если данное слово отсутствует в таблице или же его счетчик count меньше определенного значения (по умолчанию это значение равно 3, но у пользователя есть возможность поменять это значение от 1 до 10), то слово считается «неправильным» и предлагаются наиболее подходящие слова для его исправления.

Если пользователь воспользовался подсказкой от клавиатуры, то система подставит выбранное слово на место предыдущего, счетчик count выбранной подсказки увеличится на 1, орфокорректор закончит свою работу и начнет работу предиктивный ввод. Если пользователь проигнорирует подсказку и начнет дальше писать текст, то «неправильное» слово запишется в таблицу words (если оно там не находится) или увеличит его счетчик count на 1 (если оно там есть), орфокорректор закончит свою работу и начнет работу дополнение. Если пользователь нажал на символ удаления и после этого символ в конце строки не равен символу алфавита, то ничего не произойдет в плане подбора подсказок и фиксации новых слов. Иначе если при удалении курсор дошел непосредственно до самого слова (символ в конце строки равен символу алфавита), то орфокорректор закончит свою работу и начнет работу дополнение. Далее рассмотрим последний сервис — предиктивный ввод.

Предиктивный ввод позволяет предугадать следующие наиболее вероятные слова, которые могут идти за текущим словом. Модель данного лингвистического сервиса изображена в виде диаграммы деятельности (рисунок 5).

Изображение выглядит как диаграмма, схематичный

Автоматически созданное описание

Рисунок 5. Модель предиктивного ввода

Рассмотрим алгоритм. Сначала сервис проверяет на равенство предыдущее слово (слово, которое находится на данный момент перед текущим словом) с сохраненным ранее словом. Если равенство соблюдается, то данная пара слов добавляется в таблицу collocations (если ее там нет) или ее счетчик count увеличится на 1 (если она там есть). Иначе ничего не происходит. В любом случае у пользователя будут два варианта действий: либо нажать на клавишу, либо выбрать слово для продолжения из только что подобранных наиболее вероятных слов слева направо по убыванию вероятности.

Подсказки подбираются следующим образом: система обращается к данным таблицы collocations и по значению prevId текущего слова находит все пары, связанные с этим словом. С помощью счетчика count производится сортировка по убыванию. В полях подсказок оказываются самые верхние слова, чьи индексы соответствуют значениям nextId.

Если пользователь воспользовался подсказкой от клавиатуры, то на место курсора клавиатура подставит выбранную подсказку, данная пара слов добавляется в таблицу collocations (если ее там нет) или ее счетчик count увеличится на 1 (если она там есть), текущее слово сохранится в оперативную память, предиктивный ввод закончит свою работу и начнет работу дополнение. Если пользователь нажал на клавишу и данный символ не равен символу алфавита, то идет возврат в начало цикла. Иначе текущее слово сохранится в оперативную память, предиктивный ввод закончит свою работу и начнет работу дополнение.

## 2.3. Пользовательское взаимодействие с клавиатурой

В данном параграфе рассмотрим макет клавиатуры (рисунок 6) и описание ее элементов.

Изображение выглядит как текст, электроника

Автоматически созданное описание

Рисунок 6. Макет клавиатуры

В таком виде клавиатура встречает пользователя в начале работы. На самом верху три ячейки, которые будут выдавать рекомендации по дополнению, исправлению и продолжению слова пользователю в процессе его работы (на данный момент подсказки выдал сервис дополнение). Далее располагаются обычные символы. Клавиша «⇑» нужна для изменения регистра символов (рисунок 7), клавиша «×» удаляет последний символ, клавиша «» меняет раскладку клавиатуры (рисунок 8), клавиша «?123» отображает цифры, специальные символы и знаки пунктуации (рисунок 9), клавиша «↲» служит функцией для переноса строки.

Изображение выглядит как текст, электроника, калькулятор, клавиатура

Автоматически созданное описание

Рисунок 7. Макет клавиатуры с верхним регистром

Изображение выглядит как текст, электроника, черный, клавиатура

Автоматически созданное описание

Рисунок 8. Макет клавиатуры с английской раскладкой

Изображение выглядит как текст, электроника, черный

Автоматически созданное описание

Рисунок 9. Макет клавиатуры с цифрами, специальными символами и знаками пунктуации

Клавиша «ABC» возвращает клавиатуру в исходное состояние (рисунок 6 или рисунок 8 в зависимости от текущей раскладки).

У пользователя есть возможность самому задавать цветовое обозначение текста слов-дополнений, слов-коррекций и слов-продолжений. По умолчанию, слова дополнения будут темно-голубого цвета (все вышеперечисленные рисунки, то есть с 6 по 9), слова-коррекции будут зеленого цвета (рисунок 10), а слова-продолжения будут синего цвета (рисунок 11).

Изображение выглядит как текст, электроника, клавиатура

Автоматически созданное описание

Рисунок 10. Макет клавиатуры со словами-коррекциями

Изображение выглядит как текст, электроника, клавиатура

Автоматически созданное описание

Рисунок 11. Макет клавиатуры со словами-продолжениями

# 3. Реализация сервиса клавиатуры

## 3.1. Модуль серверной части

Модуль серверной части написан на языке программирования Java. Для его изучения использовалась документация [-] и сайт [-]. Java — строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems. Java превратилась из просто универсального языка в целую платформу и экосистему, которая объединяет различные технологии, используемые для целого ряда задач: от создания десктопных приложений до написания крупных веб-порталов и сервисов. Достаточно вспомнить популярность мобильной ОС Android, большинство программ для которой пишутся именно на Java.

Для реализации экранной клавиатуры был использован фреймворк Spring. Для его изучения использовалась документация [-] и сайт [-]. Spring — один из самых популярных фреймворков для создания веб-приложений на Java. Spring предоставляет большую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

В качестве системы управления базами данных была использована PostgreSQL. Для ее изучения использовалась документация [-] и сайт [-]. Данная СУБД является открытой, отличается расширенным функционалом и высоким уровнем оптимизации выполнения запросов.

В проекте имеется каталог controller, в котором принимаются запросы от клиентской части. В нем есть три класса: UsersController, CollocationsController и WordsController. Принцип работы у них похож, поэтому рассмотрим WordsController.

Класс WordsController отвечает за получение и отправку данных о словах, взаимодействуя с базой данных. Рассмотрим следующий листинг кода.

@RequestMapping(value = "", method = RequestMethod.GET)  
WordResource[] getAll(@RequestParam(required = false) Integer userId,  
 @RequestParam(required = false) Object expand)  
{  
 Word[] wordEntities = userId == null ?  
 wordsRepository.select() :  
 wordsRepository.selectByUserId(userId);  
 return Arrays.stream(wordEntities)  
 .map(entity ->  
 {  
 WordResource wordResource = new WordResource(entity);  
 if (expand != null)  
 {  
 wordResource.setUserResource(new UserResource(usersRepository.select(entity.getUserId())));  
 wordResource.setCollocationResources(Arrays.stream(collocationsRepository  
 .selectByPrevIdNextId(entity.getId(), entity.getId()))  
 .map(e -> new CollocationResource(e))  
 .toArray(CollocationResource[]::new));  
 }  
 return wordResource;  
 }).toArray(WordResource[]::new);  
}

Представленный отрывок кода отвечает за получение данных о словах из базы данных, так как является GET-методом (method = RequestMethod.GET). Здесь реализуются 4 варианта запроса:

1. Получение всех слов (userId == null и expand == null);
2. Получение всех слов с привязкой данных из других таблиц, относящихся к данным словам (userId == null и expand != null);
3. Получение всех слов по указанному идентификатору пользователя (userId != null и expand == null);
4. Получение всех слов по указанному идентификатору пользователя с привязкой данных из других таблиц, относящихся к данным словам (userId != null и expand != null).

Само обращение к базе данных реализуется через методы select и selectByUserId. Возвращаются данные в виде массива WordResource. Сам тип данных WordResource является JSON-представлением, так как реализует интерфейс Serializable.

## 3.2. Модуль клиентской части

Модуль клиентской части также написан на языке программирования Java. Про данный язык программирования подробно рассказывалось выше.

Для реализации этого модуля была использована библиотека Retrofit 2. Для его изучения использовалась документация [-] и сайт [-]. Retrofit — это REST клиент для Java и Android. Он позволяет легко получить и загрузить JSON (или другие структурированные данные) через веб-сервис на основе REST. В Retrofit можно настроить, какой конвертер использовать для сериализации данных. Обычно для JSON используется GSON, но можно добавлять собственные конвертеры для обработки XML или других протоколов. В Retrofit используется библиотека OkHttp для HTTP-запросов. В Retrofit 2, в отличие от Retrofit 1.9, не нужно явно определять OkHttp как зависимость для проекта, если нет конкретных требований к версии. Однако для Retrofit 2 требуется явно указать зависимость для автоматической конвертации данных в какой-либо формат, например, в JSON.

В качестве системы управления базами данных на локальном уровне была использована SQLite. Для ее изучения использовалась документация [-] и сайт [-]. SQLite — компактная встраиваемая СУБД. Это значит, что большинство СУБД являются самостоятельными приложениями, взаимодействие с которыми организовано по принципу клиент-сервер. SQLite является написанной на языке C библиотекой, которую динамически или статически подключают к программе.

В проекте имеется каталог clients, в котором объявлены интерфейсы для взаимодействия с API: UserClient, CollocationClient и WordClient. Приведем пример описания GET-метода интерфейса CollocationClient.

String res = "collocations";

@GET(res)  
Call<Collocation[]> get(@Query("userId") Integer userId, @QueryName Object expand);

В данном отрывке кода объявлен метод запроса get, который имеет «collocations» URL-адрес конечной точки API и возвращает объект обобщенного класса Call, имеющим в качестве своего универсального параметра массив Collocation. Классы Collocation, Word и User содержат необходимые свойства для сопоставления данных ответа. Здесь также предоставляется возможность передать userId и expand параметры запроса для получения желаемого ответа.

Само конкретное использование данных методов взаимодействия с API наблюдается при авторизации и регистрации пользователя, а также при синхронизации данных. Рассмотрим на примере синхронизации данных между таблицами collocations GET-метод в следующем листинге кода.

private void getCollocationsFromApi(Integer userId, Object expand) {  
 mCollocationClientImpl.setCallGet(userId, expand);  
 mCollocationClientImpl.getCallGet().enqueue(new Callback<Collocation[]>() {  
 @RequiresApi(api = Build.VERSION\_CODES.N)  
 @Override  
 public void onResponse(Call<Collocation[]> call, Response<Collocation[]> response) {  
 if (response.isSuccessful()) {  
 mTmpCollocationsFromApi = new ArrayList<>(Arrays.asList(response.body()));

synCollocations();

// . . .  
 }  
 else {  
 setErrorMessage(Constants.ERROR\_TRANSFER\_DATA);  
 }  
 }  
 @Override  
 public void onFailure(Call<Collocation[]> call, Throwable t) {  
 setErrorMessage(Constants.ERROR\_CONNECTION);  
 }  
 });  
}

Представленный отрывок кода отвечает за получение данных о парах слов со стороны сервера. После выполнения запроса в условии проверяется корректность его выполнения, то есть определяется код возврата. Если код удовлетворительный (например, 200), то процесс синхронизации продолжается, иначе отображается ошибка о проблеме с передачей данных. При отсутствии связи с API (например, нет доступа к Интернету) срабатывает метод onFailure, который также выводит ошибку пользователю о проблеме с соединением. Данный метод конкретно взаимодействует с методом getAll сервлета CollocationsController, расположенном на сервере. Здесь реализуются 4 варианта запроса:

1. Получение всех пар слов (userId == null и expand == null);
2. Получение всех пар слов с привязкой данных из других таблиц, относящихся к данным парам слов (userId == null и expand != null);
3. Получение всех пар слов по указанному идентификатору пользователя (userId != null и expand == null);
4. Получение всех пар слов по указанному идентификатору пользователя с привязкой данных из других таблиц, относящихся к данным парам слов (userId != null и expand != null).

Само обращение к базе данных реализуется внутри метода synCollocations, который вызывается при успешном выполнении запроса. В нем вызывается метод selectCollocations, который возвращает данные в виде списка Collocation.

## 3.3. Модуль взаимодействия

В проекте Android Studio все расположено на своих местах: исходный код в одной директории, ресурсы — в другой. В ресурсах хранятся разметки активностей, изображения, цвета, стили, строки, разметки клавиатуры и так далее. Остановимся подробней на разметках клавиатуры, так как все взаимодействие пользователя с клавиатурой, прежде всего, зависит от них.

Есть три файла с разметкой клавиатуры формата .xml: с русской, английской и прочими символами. Рассмотрим небольшой фрагмент русской разметки в следующем листинге.

<?xml version="1.0" encoding="utf-8"?>  
<Keyboard xmlns:android="http://schemas.android.com/apk/res/android"  
 android:keyWidth="7.5%p"  
 android:horizontalGap="0.833%"  
 android:verticalGap="0dp"  
 android:keyHeight="40dp">

<!-- . . . -->  
 <Row>  
 <Key android:codes="-1" android:keyIcon="@drawable/ic\_keyboard\_capslock\_white\_24dp"  
 android:keyWidth="13%p" android:horizontalGap="0%" android:keyEdgeFlags="left" />  
 <Key android:codes="1103" android:keyLabel="я" />  
 <Key android:codes="1095" android:keyLabel="ч" />  
 <Key android:codes="1089" android:keyLabel="с" />  
 <Key android:codes="1084" android:keyLabel="м" />  
 <Key android:codes="1080" android:keyLabel="и" />  
 <Key android:codes="1090" android:keyLabel="т" />  
 <Key android:codes="1100" android:keyLabel="ь" />  
 <Key android:codes="1073" android:keyLabel="б" />  
 <Key android:codes="1102" android:keyLabel="ю" />  
 <Key android:codes="-5" android:keyIcon="@drawable/ic\_backspace\_white\_24dp"  
 android:keyWidth="11%p" android:keyEdgeFlags="right" android:isRepeatable="true" />  
 </Row>  
 <!-- . . . -->  
</Keyboard>

Разберем по частям. Начинается все с главного тега <Keyboard>, в котором и расположен весь код. Атрибут android:keyWidth отвечает за ширину каждой клавиши, остальные три отвечают за горизонтальное расстояние между клавишами, вертикальное расстояние между клавишами и за высоту клавиши соответственно. Тег <Row> определяет один ряд клавиатуры (всего их 4). Мы рассмотрим третий ряд. В нем тег <Key> отвечает за одну клавишу. Через соответствующие атрибуты задается код символа и сам символ, который отображается пользователю. Клавиши изменения регистра и удаления вместо текста используют изображения, ассоциирующие их функциональность, благодаря атрибуту android:keyIcon.

Чтобы пользователь мог написать что-либо в текстовом поле, необходимо определить слушатель — функцию, обрабатывающую события при взаимодействии с пользователем, такие как нажатие на клавишу и так далее. Рассмотрим обработчик события нажатия на клавишу — функцию onKey.

public void onKey(int primaryCode, int[] ints) {  
 InputConnection ic = getCurrentInputConnection();  
 // . . .  
 if (IMESettingsStore.imeSettings.getSound() == Constants.TRUE)  
 playClick(primaryCode);  
 if (IMESettingsStore.imeSettings.getVibration() == Constants.TRUE)  
 vibrate();  
 switch (primaryCode) {  
 case Keyboard.KEYCODE\_SHIFT:  
 handleShift();  
 break;  
 case Keyboard.KEYCODE\_DONE:  
 ic.sendKeyEvent(new KeyEvent(KeyEvent.ACTION\_DOWN, KeyEvent.KEYCODE\_ENTER));  
 break;  
 case Keyboard.KEYCODE\_ALT:  
 handleSymbolsSwitch();  
 break;  
 case Keyboard.KEYCODE\_MODE\_CHANGE:  
 handleLanguageSwitch();  
 break;  
 default:  
 if (primaryCode == Keyboard.KEYCODE\_DELETE) {  
 ic.deleteSurroundingText(1, 0);  
 // . . .  
 }  
 else {  
 char code = (char) primaryCode;  
 if (Character.isLetter(code) && mIsCapsOn) {  
 code = Character.toUpperCase(code);  
 }  
 ic.commitText(String.valueOf(code), 1);  
 // . . .  
 }  
 // . . .  
 break;  
 }  
}

Вкратце рассмотрим данный код. Метод принимает код нажатой клавиши (primaryCode). Сначала получаем соединение с текущим текстовым полем, в котором мы пишем, чтобы в дальнейшем с ним взаимодействовать. Далее, исходя из настроек пользователя, ставим звук и вибрацию клавиши. Следом идет проверка на то, какую функцию выполняет нажатая клавиша, например, печатает символ, удаляет символ, меняет регистр, раскладку и так далее. Исходя из этого, на каждую функцию предусмотрено определенное действие. Например, в случае обычного символа срабатывает метод commitText, а в случае удаления — deleteSurroundingText.

# 4. Тестирование сервиса-клавиатуры

## 4.1. Тестирование модуля серверной части

## 4.2. Тестирование модуля клиентской части

Для тестирования данного модуля был использован фреймворк Espresso. Для его изучения использовалась документация [-] и сайт [-]. Espresso — это фреймворк для автоматизированного тестирования, с помощью которого можно писать лаконичные, красивые и надежные тесты пользовательского интерфейса Android. При тестировании данного модуля будет отслеживаться изменение в поведении лингвистических сервисов в зависимости от изменения базы данных. Весь процесс будет виден обычному пользователю.

В качестве тестирования рассмотрим обучение клавиатуры под определенного пользователя на примере одного слова. Слово должно быть «не знакомо» клавиатуре — его не должно быть в базе данных.

На этот раз используем в действии английскую раскладку и возьмем для тестирования слово «access». Также для быстрого получения необходимого результата ускорим обучаемость клавиатуры.

В Espresso для того, чтобы получить элемент и совершить над ним действие, можно ограничиться одной строкой, в чем его большое преимущество. Рассмотрим небольшой пример.

onView(withId(R.id.editText)).perform(click()).check(matches(isDisplayed()));

Здесь с помощью метода onView находим по id текстовое поле и сразу нажимаем на него. Также с помощью метода check проверяем, отобразилось ли текстовое поле на экране смартфона. При вводе двух первых символов слова «access» («ac») в первый раз, как видно из рисунка ---, клавиатура задействовала лингвистический сервис дополнение, который выдал три слова на выбор для завершения текущего начатого слова.

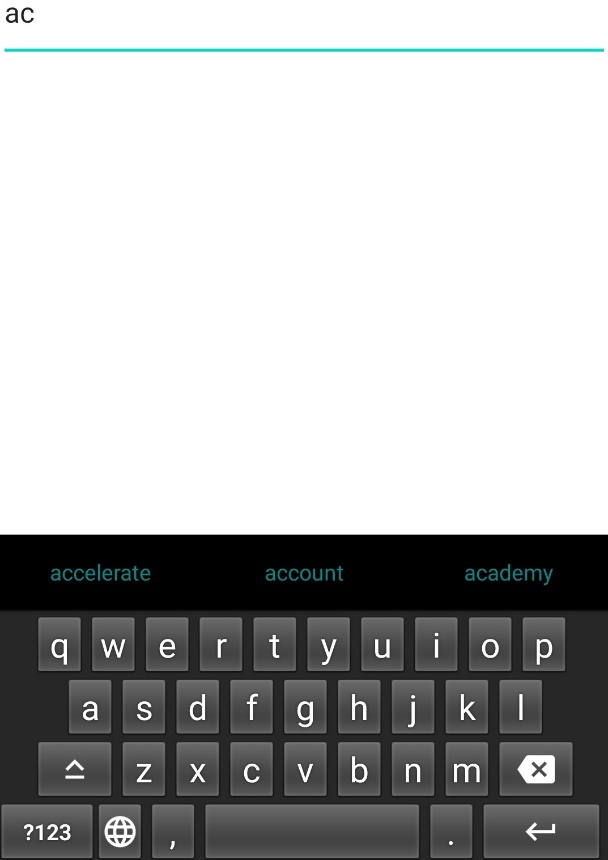


Рисунок ---. Работа дополнения

Самостоятельно завершив слово и нажав пробел, клавиатура задействовала лингвистический сервис орфокорректор, который выдал три слова на выбор для исправления (рисунок ---).

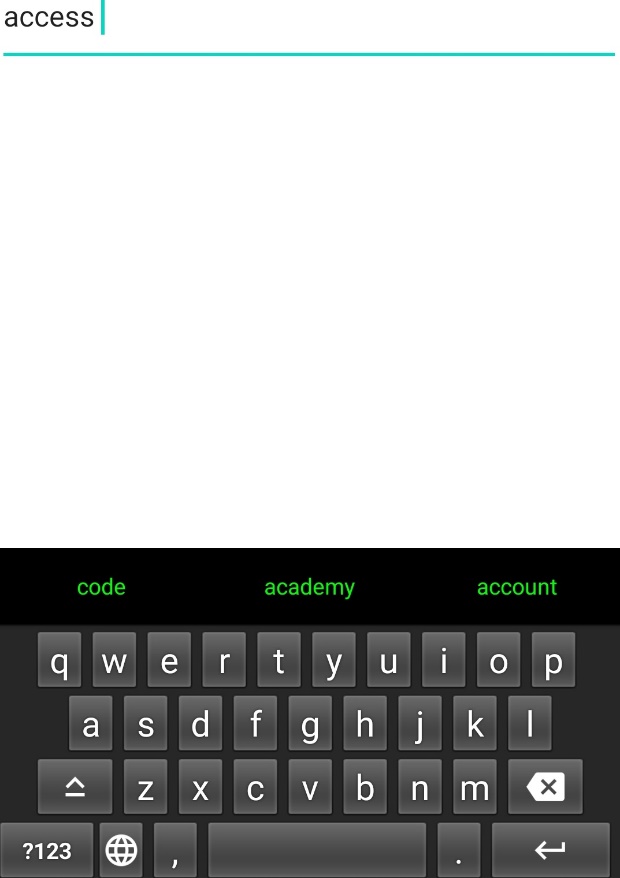


Рисунок ---. Работа орфокорректора

Продолжив печатать данное слово еще несколько раз, не исправляя его, увидим, что клавиатура запомнила текущее слово и больше не предлагает подсказки для его корректировки. На данном этапе клавиатура задействует лингвистический сервис предиктивный ввод, который будет предлагать данное слово в качестве слова-продолжения (рисунок ---).

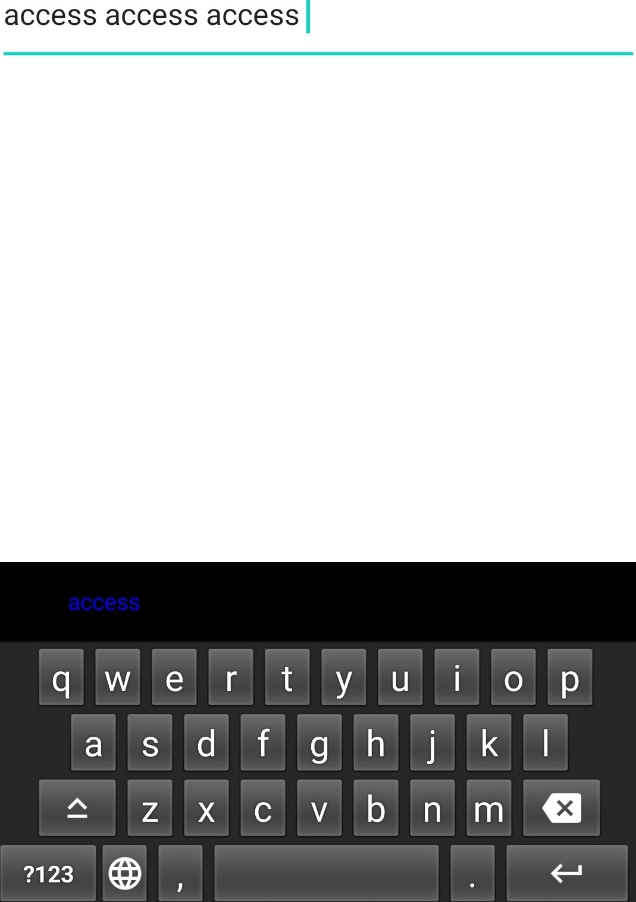


Рисунок ---. Работа предиктивного ввода

## 4.3. Тестирование модуля взаимодействия

# Заключение

# Список использованных источников