Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Прикладной информатики**

Направление подготовки **45.03.04 Интеллектуальные системы в гуманитарной сфере**

Образовательная программа **Языковые модели и искусственный интеллект**

К У Р С О В О Й П Р О Е К Т

Тема: «Разработка сервиса локализации мобильного приложения»

Обучающийся: Данилевский Алексей Александрович К3160

Дата: 05.01.2025

Санкт-Петербург 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На данный момент мобильные приложения представляют собой большой сегмент IT индустрии. Миллионы людей по всему миру ежедневно используют приложения для работы, развлечений, общения, покупок и получения услуг. Однако для успешного выхода на международные рынки разработчики сталкиваются с необходимостью локализации своих продуктов. Локализация приложения призвана адаптировать интерфейс, текстовое наполнение, культурные аспекты и функционал к особенностям конкретного региона или языка. При этом на рынке отсутствуют бесплатные решения, что побуждает его участников либо создавать свои сервисы локализации, либо решать классическую проблему соотношения цены и качества.

Целью нашего проекта является разработка бесплатного и независимого сервиса локализации мобильных приложений, который помогает адаптировать его язык, формат содержание для удобства пользователей разных стран и культур.

Проект был разделен на следующие подзадачи:

– Обучить модель,
– Разработать API,
– Разместить облачный сервер,
– Разработать MVP.

1 Описание проекта

Проект представляет собой платформу для локализации мобильных приложений, обеспечивающую автоматизацию и упрощение процесса адаптации приложений для международных рынков, адаптируя для разработчиков и издателей их продукты под различные языки целевых рынков, минимизируя затраты времени и ресурсов.

Проект был построен на основе передовых технологических решений, которые обеспечивают потенциальную универсальность и в частности высокую производительность

Более подробное описание используемых технологий:

– OPUS Books, Kazparc, WMT14, Wlhb/Translation представляют собой комплекс взаимосвязанных технологий и платформ, ориентированных на обработку и преобразование текстовой информации. В проекте использовались в качестве материала для тренировки языковой модели;

– ISBN DB – база данных, содержащая информацию об ISBN (International Standard Book Number) книг, включая их названия, авторов, издательства и другие метаданные;

– DeeplAPI – API, предоставляемый компанией Deepl, которая специализируется на машинном переводе. Это API позволяет выполнять перевод текстов с одного языка на другой;

– Google T5 (Text-to-Text Transfer Transformer) – модель автоматического обучения, разработанная компанией Google, которая может решать широкий спектр задач, связанных с обработкой естественного языка;

– Marian – предоставляет эффективные и высокопроизводительные алгоритмы для обучения и развертывания моделей машинного перевода;

– SQLAlchemy – python-библиотека для работы с базами данных. Она предоставляет высокоуровневый, объектно-ориентированный интерфейс для взаимодействия с базами данных;

– Psycopg2 – драйвер Python для базы данных PostgreSQL. Это позволяет SQLAlchemy работать с PostgreSQL, одной из самых популярных реляционных баз данных;

– PyTorch – open-source библиотека машинного обучения, созданная компанией Facebook AI Research. Она предоставляет высокоуровневые инструменты и API для создания, обучения и развертывания моделей глубокого обучения;

– PostgreSQL – объектно-реляционная система управления базами данных с открытым исходным кодом;

– Flutter – Кроссплатформенный фреймворк для разработки мобильных приложений, использующий язык программирования Dart;

– Dart – Динамический, объектно-ориентированный язык программирования, разработанный для создания мобильных, веб- и серверных приложений;

– Android Studio – итнтегрированная среда разработки для разработки Android-приложений, созданная компанией Google;

– Swift UI – декларативный фреймворк для создания пользовательских интерфейсов на iOS, iPadOS, macOS, tvOS и watchOS;

– Xcode – комплексная интегрированная среда разработки, разработанная Apple для создания приложений для различныз устройств Apple.

2 Описание задачи, поставленной передо мной

Основная проблема, которую мне было необходимо решить, это предоставить возможность моей команде наглядно продемонстрировать работу языковой модели для людей, не специализирующихся в данной области. Нужно было показать, каким именно образом можно реализовать интеграцию сервиса локализации на какое-то мобильное приложение. При этом, желательно было выбрать такое мобильное приложение, которое требует перевода. Таким образом, мне было необходимо создать прототип мобильного приложения, похожего на стандартное приложение стандартного онлайн маркетплейса, на котором можно применить нашу модель.

3 Ход работы

До начала работы ни с одной из технологий, кроме Visual Studio code я знаком не был. Поэтому первым делом, я начал изучать информацию по всем технологиям.

В самом начале работы над проектом, ввиду отсутствия у меня опыта, руководителем проекта было предложено два варианта решения поставленной задачи: или использовать платформы разработки без кода, или фреймворк Flutter. С целью получения большего количества опыта и знаний я пошел по второму пути.

Решение задачи проводилось в несколько этапов, несмотря на ее принципиальную простоту. Изначально было решено сделать две версии, для системы Android и iOS. Для первой был выбран стек технологий в виде языка программирования Dart, фреймворк разработки приложений Flutter, среда разработки VS code, а также Android studio. Для второй, соответственно, Swift, SwiftUI, Xcode, а также Swift PM.

Сперва я решил заняться Android версией. Для этого я начал изучать фреймворк Flutter, таким образом одновременно осваивая язык Dart. Этот процесс включал в себя чтение официальной документации а также просмотра видеоуроков из сети. После изучения основ, я решил продолжить изучение параллельно практикуясь и создавая уже итоговый продукт. Тут следует отметить, что на первых этапах я активно использовал генерацию кода искусственным интеллектом, что позволило ускорить процесс понимания работы фреймворка и языка.

Таким образом относительно последовательно были реализованы начальный экран с сеткой карточек товаров, что проиллюстрировано на рисунке 1.

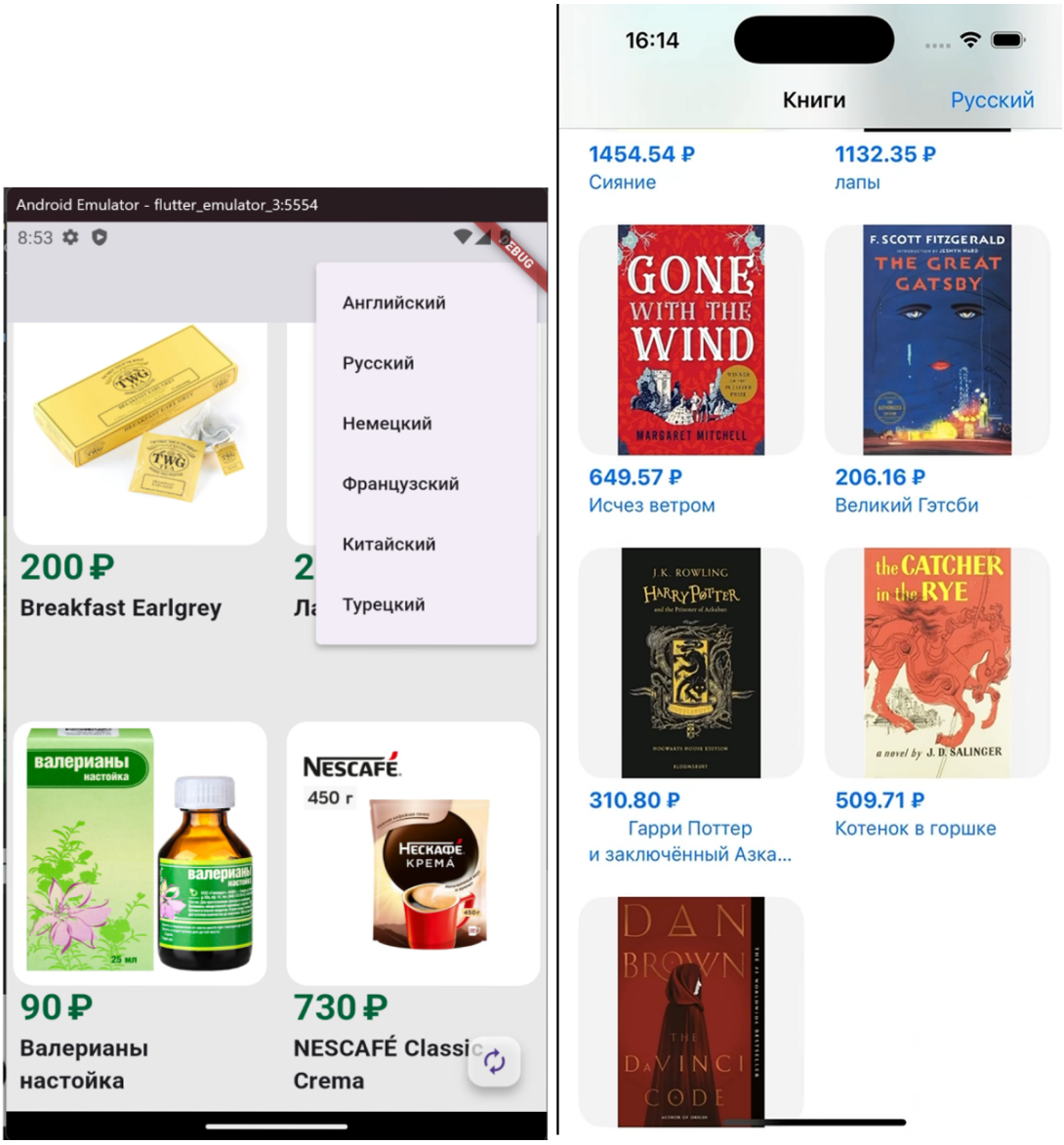

Рисунок 1 — изображение начального экрана приложения, слева – Android версия, справа – iOS

Страница с детализированным описанием отдельной карточки, навигация между страницами, проиллюстрировано на рисунке 2.

Рисунок 2 — страница с детализацией карточки товара, слева – Android версия, справа – iOS

Далее была реализована связь с бэкэндом, в виде запроса набора товаров, и парсинга получаемых данных из формата, установленного по контрактус бэкендом. Требуемый формат изображен на рисунке 3, реализация парсинга данных – на рисунке 4.

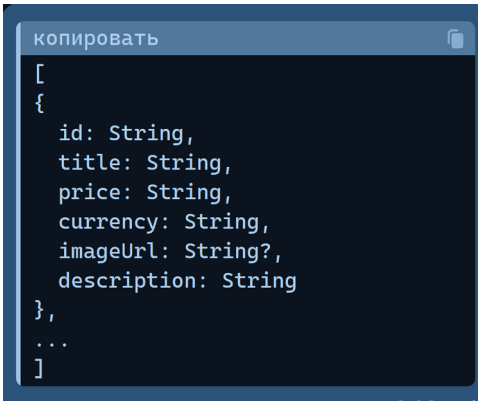

Рисунок 3 — формат JSON файла, установленный по контракту

```dart
class Item {
  final String id;
  final String title;
  final String price;
  final String currency;
  final String description;
  final String imageUrl;

  Item({
    required this.id,
    required this.title,
    required this.price,
    required this.currency,
    required this.description,
    required this.imageUrl,
  });

  factory Item.fromJson(Map<String, dynamic> json) {
    return Item(
      id: json['id'],
      title: json['title'],
      price: json['price'],
      currency: json['currency'],
      description: json['description'],
      imageUrl: json['imageUrl'],
    );
  }
}

Future<List<Item>> getItemsList_1(Locale) async {
  final response = await http.get(
      Uri.parse(
          'https://raw.githubusercontent.com/Poleksey/localization_mvp/refs/heads/main/demo.json'),
      headers: {'locale': Locale});

  final List<dynamic> jsonData = jsonDecode(response.body);

  return (jsonData).map((itemData) => Item.fromJson(itemData)).toList();
}
```

Рисунок 4 — иллюстрация кода декодирования принятого JSON формата в Android версии

```swift
import Foundation

public struct Product: Codable, Equatable, Hashable, Identifiable, Sendable {
    public let id: String
    let title: String
    let description: String
    let price: Double
    let currency: String
    let imageUrl: String

    private enum CodingKeys: String, CodingKey {
        case id
        case title
        case description
        case price
        case currency
        case imageUrl
    }
}

```

```swift
import Foundation
import Networking

private enum Constants {
    static let productsPath = "/products/"
}

enum Endpoint {
    case getProducts(String)
}

extension Endpoint: EndpointProtocol {

    var method: HTTPMethod {
        switch self {
        case .getProducts: .get
        }
    }

    var path: String {
        switch self {
        case let .getProducts: Constants.productsPath
        }
    }

    var headers: [String: String] {
        switch self {
        case let .getProducts(locale): ["accept": "application/json", "locale": locale]
        }
    }
}

```

Рисунок 5 — иллюстрация кода декодирования принятого JSON формата в iOS версии

На этом этапе работы было необходимо создать набор заготовленных данных для непосредственного тестирования моделей. Для

более корректного отображения была выбрана тематика “книги”. Для Android версии, ввиду некоторых проблем, описанных ниже, использовался другой набор данных. Для первоначального тестирования сетевого слоя вместо для упрощения выполнения этапа использовалась технология GitHub Raw.

Вышло немного непоследовательно, но под конец работы над Android версией, я узнал о различных типах архитектуры мобильных приложений. То, что я делал произвольно, было похоже на один из них (MVVM), поэтому я решил подогнать имеющийся материал под эту архитектуру.

Так как масштаб Android версии были крайне незначительными, реализация iOS версии по большей части представляла собой переписывание кода с Android версии на язык Swift с сохранением архитектуры, за исключением того, что к сетевому слою по рекомендации руководителя был применен Swift PM , то есть он был упакован в пакет для удобного переиспользования. Кроме того, значительно проще было проводить парсинг данных, ввиду наличия протоколов, перечисленных на третьей строке первой части рисунка 5: Hashable - этот протокол позволяет использовать объекты этой структуры в качестве ключей в словарях или множествах; Identifiable - этот протокол требует, чтобы структура имела свойство id типа String, которое служит уникальным идентификатором для каждого объекта. Эти протоколы в итоге позволяют избежать подбора комбинаций декодирования файлов JSON формата. Также ввиду особенностей фреймворков визуального окружения, использующихся для разных версий, внешний вид мобильных приложений получился немного разным, что можно увидеть на рисунках 1 и 2.

4 Что в рамках проекта сделать не получилось

Были вещи, которые завершить не удалось. Здесь следует отметить, что разработка разных версий делалась на разных устройствах. Android версия – на системе Windows. В связи с этим на финальном этапе совмещения фронтенда и бэкенда возникли неустранимые проблемы (на системе отсутствовал компонент WSL, что сделало невозможным запустить Docker контейнер, а следовательно протестировать работу языковой модели). Кроме того, все остальные проблемы по большей части были связаны именно с системой Windows. Очень много времени ушло на настройку всех компонентов рабочего окружения. Например, пришлось постараться с тем, чтобы снять заводские ограничения на установку переменных среды окружения таким образом, чтобы они работали на всей системе, несмотря на то, что ранее подобные действия уже получалось реализовать с другими компонентами (Python). Также в Android версии много времени ушло на разработку системы парсинга данных по по схеме заказчика, ввиду несколько запутанного синтаксиса пакетов языка Dart в данной области технологий.

5 О ходе работы над проектом

Несмотря на общую не зависящую от проекта высокую нагрузку работать удалось планомерно. В зависимости от ситуации у меня получалось выделять на проект время раз в одну или две недели. Рабочие сессии представляли собой примерно одинаковый процесс на протяжении всей работы над проектом, включая в себя изучение части материала теории, и его реализацию, таким образом, изначально можно было рассчитать, сколько таких рабочих сессий мне понадобиться до конца проекта.

6 Приобретенный в рамках проекта опыт

В ходе выполнения курсового проекта мне пришлось и удалось познакомится, учитывая небольшое количество времени, можно сказать, со многими технологиями. Среди них Android studio непосредственно, а также настройка данной технологии и интеграция ее с основной средой разработки, также я закрепил на практике базовые навыки работы с системой контроля версий Git, в ходе непосредственной работы познакомился со встроенными базовыми библиотеками фреймворка Flutter и языком Dart, в частности пришлось работать с ООП, функциями, настройкой виджетов, впервые встретился и изучил на практике JSON формат файлов, а также архитектуру мобильных приложений, в меньшей степени ознакомился с соответствующими аналогами для разработки iOS версии. Помимо перечисленного, что важно, я осознал неполноценность Windows OS как рабочей операционной системы, что побудило меня уже после выполнения проекта перейти на GNU/Linux, что само по себе дает множество преимуществ для меня в дальнейшем, не говоря о возможностях для развития.

7 Коммуникация в команде

7.1 О взаимодействии с командой

О взаимодействии с командой. Много взаимодействовать с командой у меня необходимости не было, так как большая часть работы не требовала синхронизации с действиями остальных. Но на этапе развертывания сетевого слоя пришлось корректировать действия части команды, отвечающей за бэкенд: нужно было получить информацию JSON формата в нужной структуре данных, как того требовал руководитель.

7.2 О взаимодействии с руководителем проекта

О взаимодействии с руководителем проекта. Взаимодействие с руководителем проекта вышло получилось очень продуктивным. В первую очередь, Игорь Манаков помог отобрать материалы для изучения Flutter, что, полагаю, позволило сэкономить существенное количество времени, за что я ему очень благодарен. Кроме того, руководитель помог разобраться с профессиональной терминологией, с которой я до этого проекта не встречался. Также были очень ценны комментарии к некоторым технологиям, основанные на его личном рабочем опыте.

Исходя из всего вышеперечисленного, отсутствия каких-либо серьезных недочетов и гармоничной коммуникации считаю, что мой руководитель заслуживает высшей оценки.

ЗАКЛЮЧЕНИЕ

В заключение можно отметить несколько положений. Во-первых, считаю, что поставленная цель проекта выполнена. У команды получилось создать удовлетворяющий требованиям технического задания продукт, который можно интегрировать в мобильные приложения.

Несмотря на несущественные проблемы с Android версией, решающиеся переходом на другую операционную систему и повторную настройку рабочего окружения, считаю глобально свою задачу выполненной, так как в итоге iOS версия позволила протестировать и продемонстрировать работу модели.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Официальная документация Flutter https://flutter.dev/learn

2 Статья о декодировании JSON файлов https://medium.com/@dudhatkirtan/how-to-decode-json-in-flutter-ultimate-guide-2024-b2869588d52b

3 Видеокурс по базовым элементам Flutter https://www.youtube.com/watch?v=pTJJsmejUOQ

4 Статья об архитектуре MVVM https://habr.com/ru/articles/776344/

5 Официальная документация SwiftUI https://developer.apple.com/xcode/swiftui/

ПРИЛОЖЕНИЕ (ТЕХНИЧЕСКОЕ ЗАДАНИЕ)

1. **Название:** Разработка сервиса локализации мобильного приложения

2. **Цель (назначение):** Разработать MVP сервиса для локализации, который может принимать нужные строки приложения, а затем выводить их переведёнными на нужные языки.

3. **Сроки выполнения:** Начало – 31.10.2024 Окончание – 18.12.2024

4. **Исполнитель проекта (руководитель проекта):** Игорь Манаков

5. **Термины и сокращения:**

MVP (minimal viable product) – продукт, обладающий минимальными, но достаточными для удовлетворения первых потребителей функциями

Параллельный корпус – большие собрания текстов с выравниванием по предложениям с сопоставлением одного языка с другим.

Датасет – коллекция табличных данных

Loss (Функция потерь) – функция, характеризующая потери при неправильном принятии решений на основе наблюдаемых данных

6. **Технические требования:**

- сервис должен уметь переводить тексты асинхронно на выбранных языках
- сервис должен базироваться на удалённом сервере
- сервис должен поддерживать хостинг картинок
- сервис должен иметь API для доступа к модели

7. **Содержание работы**

Таблица 1 – Содержание работы

№	Этап плана	Сроки выполнения	Ответственный за этап
1	Собрать данные	31.10.2024 – 06.11.2024	Калинин Марк Алексеевич
2	Изучить FastAPI	31.10.2024 – 10.11.2024	Ахмедов Бахадыр Бахтиёрович

3	Изучить Docker	31.10.2024 – 10.11.2024	Толкачёв Тимур Сергеевич
4	Познакомиться с Hugging Face	31.10.2024 – 10.11.2024	Скворцов Денис Александрович
5	Обработать данные	10.11.2024 – 17.11.2024	Скворцов Денис Александрович
6	Создать интерфейс для интеграции с backend	10.11.2024 – 17.11.2024	Калинин Марк Алексеевич
7	Добавить перевод навигационных компонентов	17.11.2024 – 20.11.2024	Калинин Марк Алексеевич
8	Разработать Endpoint’ы с конфигурацией мобильного	10.11.2024 – 24.11.2024	Ахмедов Бахады
9	Развернуть сервер	10.11.2024 – 24.11.2024	Толкачёв Тимур Сергеевич
10	Оптимизировать модель	17.11.2024 – 01.12.2024	Скворцов Денис Александрович
11	Обучить модель	20.11.2024 – 01.12.2024	Калинин Марк Алексеевич
12	Интегрироваться в сервис	24.11.2024 – 01.12.2024	Толкачёв Тимур Сергеевич
13	Разработать контракт API	24.11.2024 – 01.12.2024	Ахмедов Бахадыр Бахтиёрович
14	Сверстать UI мобильного приложения	31.10.2024 – 08.12.2024	Данилевский Алексей Александрович
15	Сгенерировать документацию API	01.12.2024 – 08.12.2024	Ахмедов Бахадыр Бахтиёрович
16	Доработать модель	01.12.2024 – 20.12.2024	Скворцов Денис Александрович
17	Добавить кэширование данных	01.12.2024 – 20.12.2024	Толкачёв Тимур Сергеевич
18	Интегрировать S3	08.12.2024 – 20.12.2024	Ахмедов Бахадыр Бахтиёрович
19	Создать сервисный слой приложения	08.12.2024 – 20.12.2024	Данилевский Алексей Александрович