Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Работа допущена к защите
Директор ВШ ПИ
_____ П.Д. Дробинцев
«_»_____ 2019 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

«Разработка алгоритма оценки продуктивности по различным метрикам на примере приложения-планировщика»

по направлению подготовки 09.03.04 «Программная инженерия»
по образовательной программе 09.03.04_1 «Технология разработки и сопровождения качественного программного продукта»

Выполнил студент гр. 3530904/80101 <подпись> Сопрачев А. К.

Руководитель, старший преподаватель <подпись> Селин И. А.

Санкт-Петербург
2022

РЕФЕРАТ

ABSTRACT

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире цифровые технологии упрощают и улучшают всё больше аспектов жизни человека, уже невозможно представить наш быт без мессенджеров, видеозвонков, покупок на онлайн маркетплейсах и интерактивных карт, способных в считанные секунды проложить маршрут из одного конца говора в другой. Все эти действия позволяют сделать смартфоны, лежащие в кармане практически у каждого жителя мегаполиса.

Однако пожелав посетить научную конференцию Яндекса, посвящённую передовым технологиям, я был вынужден искать конференц-зал по редким указателям, а также воспользоваться помощью сотрудников, подсказывающих куда именно надо идти. С аналогичной проблемой сталкивается большинство людей пытающихся найти конкретное место в незнакомом для них помещение, будь то информационная стойка аэропорта или учебный кабинет в одном из корпусов университета. И если крупные аэропорты с переменным успехом пытаются решать эту проблему, то студентам остаётся надеяться только на себя и непривычные аналоговые указатели в новых корпусах.

Своим проектом я хочу предложить решение этой проблемы. Целью работы является разработка мобильного приложения для платформы iOS, отображающего карту помещений и прилегающей к ним территории, позволяющего пользователю производить поиск по карте и строить маршруты до интересующих его мест. В качестве местности был выбран кампус Политехнического Университета.

В процессе разработки необходимо решить следующие задачи:

- Провести анализ предметной области и определить какие технологии для решения этой проблемы уже существуют на рынке, определить их преимущества и недостатки
- Разработать методику высоко детализированной картографии и с её по-

мощью составить план помещения и прилегающей территории

- Разработать мобильное приложения для операционной системы iOS, которое будет отображать карту и предоставлять пользователю удобный интерфейс для поиска кабинетов и построения маршрута
- Разработать возможность ”поделиться”маршрутом с помощью графических кодов

Потребность разработки карты не только для помещений, но прилегающих территорий вытекает из банальной удобности использования единого приложения. При желании построить маршрут из кабинета одного корпуса до кабинета другого, пользователю будет достаточно проложить этот маршрут в одном приложение. Кроме того, внутренняя планировка кампуса университета на сторонних картах не отличается высокой точностью и достоверностью.

Возможность поделиться маршрутом с помощью графического представления будет актуальна для организаторов мероприятий, которые могут распечатать его на приглашениях.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Актуальность

С глобальным распространением смартфонов карты стали неотъемлемой частью жизни человека, практически каждый имеет в кармане не только подробную и интерактивную карту местности, но и возможность построить маршрут до интересующей точки. Однако детализация существующих решений ограничивается автомобильными и крупными пешеходными дорогами. При этом пешеходные маршруты таких карт зачастую достаточно условны и не совпадают с реальным миром или же являются труднопроходимыми. Навигация внутри помещений в лучших случаях осталась в устаревшем аналоговом мире в виде указателей или же статичных бумажных карт. Современному человеку привыкшему к интерактивным картам бывает тяжело ориентироваться в незнакомом помещении.

1.2. Область применения

Основная область применения картографического приложения – помощь в ориентирование на незнакомой местности, в частности на территории кампуса университета. Планировка помещений позволит легко и удобно находить не только учебный корпус, но и кабинет в нём.

Сопутствующим продуктом разработки является высоко детализированная карта, глобальный спрос на которые растёт, например такие карты необходимы в сфере робототехники, например для автопилотируемых роботов-доставщиков Яндекс или же для точного позиционирования с использованием технологий дополненной реальности (AR).

(а) Ровер в университете Огайо, США

(б) Навигационное приложение в AR

Рис. 1.1: Примеры технологий которым необходимы высоко детализированные карты

1.3. Обзор существующих решений

Проект можно разделить на 3 части:

- Детализированная карта местности
- Внутренняя планировка зданий
- Построение маршрута

Рассмотрим существующие решения для каждой из частей

Детализированная карта местности

На данный момент на рынке можно выделить два популярных приложения предоставляющих пользователям высоко детализированную карту местности: Apple Maps и 2GIS

Apple Maps

Достоинства:

- Качественная высокая детализация автомобильных дорог и парковых зон
- Все примитивы карты привязаны к реальным физическим размерам. Например, дороги в классических картографических приложениях отображаются в виде линий – ребёр графа и предоставляют пользователю информацию о том, что дорога существует, в AppleMaps дороги являют-

ся многоугольниками, демонстрируя пользователям физическую ширину дороги и пространство занимаемое ей, это позволяет легче оценивать реальную проходимость дороги

Недостатки:

- Высокая детализация реализована экспериментально в нескольких городах США, картография на территории России в ближайшие 5 лет не планируется
- Акцент сделан на автомобильные дороги, пешеходные маршруты недостаточно проработаны. Например, тротуар отображается так же, как тропинка в парке

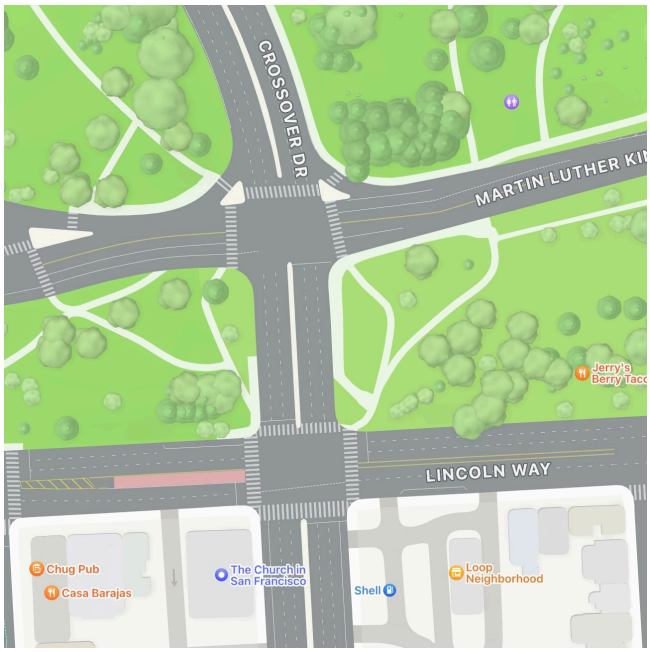

Рис. 1.2: Пример детализации местности Apple Maps

2GIS

Доступна в России. Рассматривалась на примере кампуса СПбПУ Достоинства:

- Акцент на городах России
- Наивысшая достоверность среди конкурентов

Недостатки:

- У дорог отсутствует привязка к физической ширине, из-за чего основная

аллея выглядит ровно так же, как и парковая тропинка

- Некоторые автомобильные дороги внутри кампуса отсутствуют

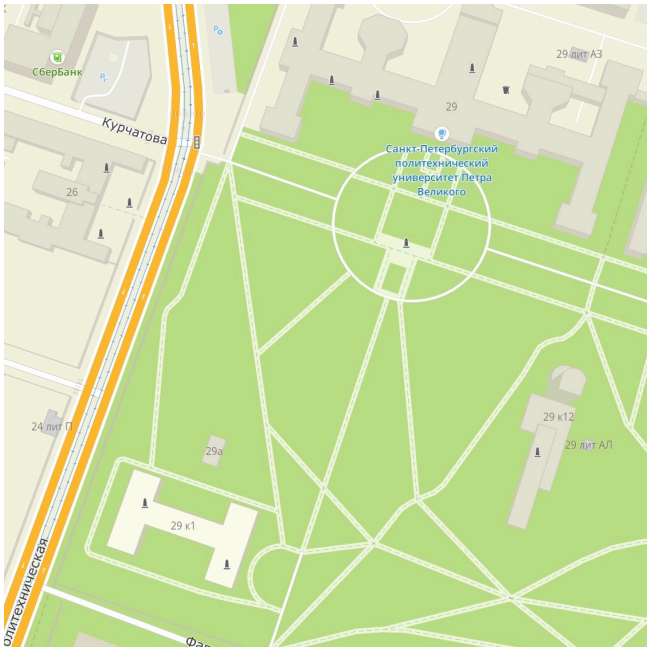

Рис. 1.3: Пример детализации местности 2GIS

Внутренняя планировка зданий

Многие компании владеющие популярным приложениями карт, разрабатывают свои решения для отображения на внутренней планировки зданий. Основным недостатком всех решения является отсутствие возможности строить маршрут внутри помещения.

Apple Maps

Достоинства:

- Наиболее детализированные планы
- Открытый универсальный стандарт описания планов помещений. Обладает подробной документацией, стандартизирован [**https://www.ogc.org/standards/requests/202**]
- Возможность добавлять свои планы в официальное приложение

Недостатки:

- Приоритет добавления своих планов отдан резидентам США

- Высокие критерии добавления своих планов (посещаемость более 1 миллиона человек в год, подходит в основном для аэропортов и вокзалов)
- При наличии публичного формата, отсутствует инструментарий для создания планов в этом формате. На рынке есть ряд компаний предоставляющих услуги картографии, однако эти компании не распространяют своё программное обеспечение

Рис. 1.4: Пример планировки зданий Apple Maps

Яндекс.Карты

Достоинства:

- Есть официальный редактор планов помещений

Недостатки:

- Закрытый проприетарный формат хранения планов помещений, отсутствует возможность экспорта в файл для использования в сторонних приложениях
- Размытая документация описывающая процесс добавления планов на карту
- На данным момент добавление планов на карту недоступно для произвольных зданий, новые схемы создаются по заявкам для торговых цен-

тров и вокзалов

Рис. 1.5: Пример планировки зданий Яндекс.Карты

2GIS Этажи

Достоинства:

- Яркий и контрастный дизайн

Недостатки:

- Карты создаются по индивидуальным заявкам
- Полностью отсутствует информация о процессе создания, требованиях для планов помещений, формате хранения

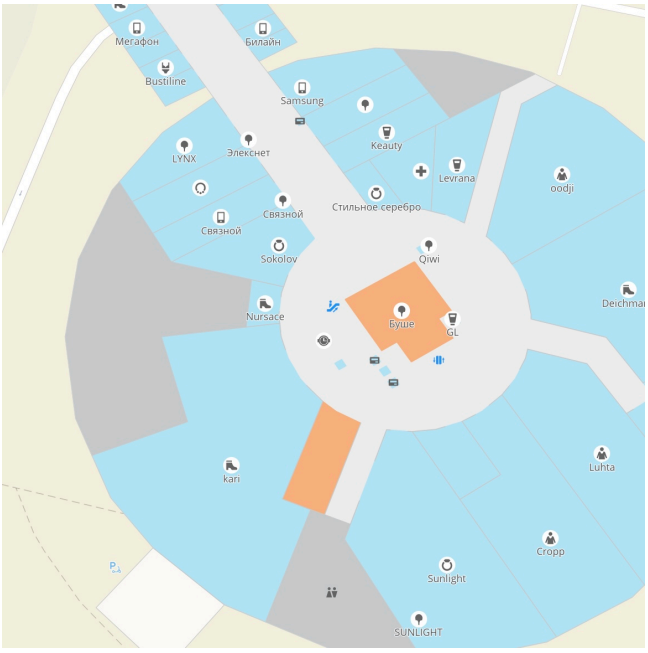

Рис. 1.6: Пример планировки зданий 2GIS Этажи

Построение маршрута

Построение маршрута по улице работает одинаково хорошо во всех популярных приложениях, однако ни одно из них не поддерживает построение маршрута внутри помещений. Существуют непубличные картографические сервисы, а так же компании разрабатывающие indoor карты под заказ с возможностью строить маршрут, однако они не вписываются в концепцию ВКР

ГЛАВА 2. ТРЕБОВАНИЯ

Перед началом разработки были составлены бизнес-требования в формате пользовательских историй, а так же конкретизированы общие требования разработки программного обеспечения промышленного качества

2.1. Общие требования к приложению

- Приложение должно не противоречить принципам дизайна Apple Human Interface Guidelines для iOS [**HumanInterfaceGuidelines**]
- Приложение должно поддерживать светлую и тёмную темы, переключение которых синхронизировано с системной темой операционной системы
- Приложение должно поддерживать все виды современных устройств на операционной системы iOS, а в частности iPhone, iPad, iPad Mini, а так же режим SplitView [**SplitView**].
- Версия приложения для планшетов должна поддерживать как портретный, так и ландшафтный режим отображения
- Приложение должно стабильно работать в стандартной частоте кадров устройства (120 или 60 кадров в секунду)
- Приложение должно поддерживать устройства с операционной системой iOS 13 и выше

2.2. Пользовательские истории

2.2.1. Для студентов

- Как студент, я хочу строить маршруты до произвольного кабинета, чтобы найти деканат
- Как студент, я хочу видеть примерное время на маршрут, чтобы заранее оценить время пути и не опоздать на пару
- Как студент, я хочу искать кабинеты через поиск по ключевым словам,

чтобы быстро найти местоположение кабинета на карте

- Как студент, я хочу видеть дополнительную информацию (адрес, телефон, почта) об объектах на карте, чтобы быстро связаться с деканатом

2.2.2. Для гостей университета

- Как гость отсканировавшый QR код в приглашение на мероприятие, я хочу чтобы открылась интерактивная карта с уже проложенным маршрутом, чтобы найти место проведения мероприятия
- Как гость пришедший на мероприятие, я хочу чтоб приложение использовало технологию AppClip, чтобы открывать его не скачивая из AppStore

2.2.3. Для организаторов

- Как организатор мероприятия, я хочу создавать красиво оформленные AppClip/QR коды с произвольным маршрутом без необходимости взаимодействовать с разработчиком, чтобы быстро и удобно создавать приглашения
- Как пользователь, я хочу быстро делиться произвольной точкой назначения с помощью ссылки, чтобы организовывать локальные встречи

ГЛАВА 3. КАРТОГРАФИЯ

При детальном рассмотрении доступных на рынке вариантов создания картографического приложения с поддержкой планов помещений, я остановился на написание своей программы, которая будет отображать карты в формате IMDF разработанным компанией Apple для Apple Maps

3.1. Формат IMDF

Формат Indoor Mapping Data Format (IMDF) [**IMDF**] является обобщённой моделью описания планировок помещений, разработан компанией Apple для внутренних нужд, после чего сделан полностью свободным и бесплатным форматом при любых условиях использования. Стандартизирован Open Geospatial Consortium в 2021 году [**https://www.ogc.org/pressroom/pressreleases/4415**]. Формат является расширением GeoJSON RFC 7946 [**GeoJSON RFC 7946**] и полностью совместим с ним.

3.1.1. Структура IMDF

Согласно нотации GeoJSON, основной структурной единицей модели является Feature, модели одинакового типа складываются в массив FeatureCollection и сохраняются в формате JSON в файл с соответствующим типу модели названием.

Базовые типы

Нотация определяет следующие базовые типы используемые во всех моделях:

LABELS

JSON объект используемый для обозначения одной лексической конструкции (строки) на разных языках. Пример:

Листинг 3.1: Пример модели LABELS

```
{
  "en": "Hello",
  "de": "Hallo"
}
```

DISPLAY-POINT

Используется для обозначения точечной координаты на карте в соответствие с нотацией GeoJSON

Листинг 3.2: Пример модели DISPLAY-POINT

```
{
  "type": "Point",
  "coordinates": [ 100.0, 0 ]
}
```

UUID

Уникальный внутри всего формата номер соответствующий GeoJSON FEATURE-ID, то есть являющийся строкой или числом. Я использую UUID v4, то есть 32х символьный номер в шестнадцатеричной системе счисления, полученный с использованием генератора случайных чисел, пример: 123e4567-e89b-12d3-a456-426614174000

RESTRICTION-CATEGORY

Описывает возможное ограничение действующие на объект

Таблица 1: Виды ограничений

Категория	Описание

employeesonly	Доступ разрешен для сотрудников имеющих пропуск
restricted	Доступ ограничен для широкой публики, однако разрешен для каких либо групп

Модели IMDF (Feature)

Вопросительный знак после типа поля означает его опциональность. Опциональные поля могут быть заполнены пустыми значениями (null) или полностью отсутствовать (undefined).

Нотация IMDF исчерпывающи описывает следующие типы моделей (Feature), подробное описание доступно в документации[**IMDFFeatures**], диаграмма сущностей в приложение **??**:

Address

Модель используется для описывания точного почтового адреса согласно местным правовым нормам

Building

Модель описывает структуру здания, при этом не описывая его физические границы, делегируя это модели Footprint.

Таблица 2: -

Поле	Тип	Описание
name	LABELS?	Официальное название здания
alt_name	LABELS?	Альтернативное название здания, которое чаще всего используется в устной речи посетителей

category	BUILDING-CATEGORY	Категория, которая лучше всего описывает функцию здания
restriction	RESTRICTION-CATEGORY?	Категория, которая лучше всего описывает ограничение, применимое ко всему физическому зданию
display_point	DISPLAY-POINT?	Координата используемая в качестве точечного представления здания
address_id	UUID?	Идентификатор модели адреса (Address) этого здания

Примером категории здания может служить ”Учебное помещение ”Паркинг ”Магазин”.

В проекте используется для описания всех видов построек, а в частности учебных корпусов, магазинов, ресторанов, трансформаторных будок и жилых домов на территории кампуса.

3.1.2. Иерархическая структура IMDF

Несмотря на то, что нотация описывает модели в нормализованном формате, для лучшего понимания, следует рассматривать их как иерархическую структуру.

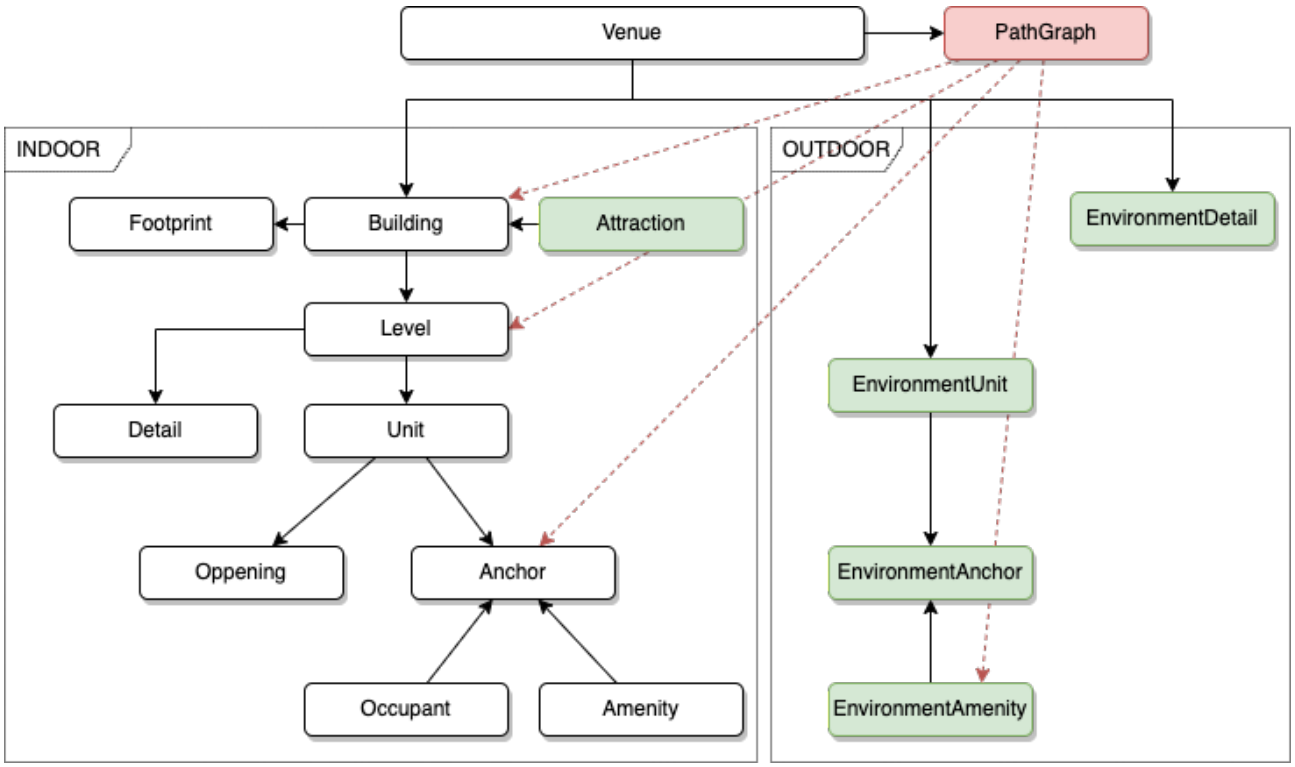

Рис. 3.1: Иерархическая структура представления данных

- Самый верхний уровень представляет модель Venue это общее пространство отображаемой карты, в случае моего приложение – весь кампус университета.
- Venue содержит в себе массив зданий (Building), форма которых описывается с помощью Footprint и на которые опционально ссылается модель Attraction, описывающая точки интереса.
- Каждое здание содержит массив этажей (Level)
- Каждый этаж содержит массив помещений (Unit) и детали (Detail)
- Каждое помещение содержит проходы (Oppening) и якоря (Anchor)
- На якоря ссылаются удобства (Amenity) и аннотации (Occupant)

3.2. Расширение стандарта

Как видно из иерархической диаграммы, верхним уровнем формата является пространство содержащие в себе здания, структура здания исчерпывающе описана, однако нотация не предоставляет моделей для описания местности между этими зданиями внутри Venue. Тем самым стандарт захватывает

часть территории, при этом позволяет описать эту территорию только зданиями, оставляя пространство между ними занятым, но пустым. Для решения этой проблемы, я решил расширить формат добавив в него модели описывающие местность между зданиями. Добавленные мной модели отмечены зелёным на диаграмме.

3.2.1. Описание добавленных моделей

EnvironmentUnit

Так как поверхность земли на улице одноэтажна, верхним уровнем можно сразу сделать модель описывающую конкретную геометрию уличных объектов.

Модель описывает все физические пространства на улице, такие как дороги, тротуары, газон, лес, тропинки. Аналогично комнатам Unit, модель EnvironmentUnit

Таблица 3: Модель уличного пространства

Поле	Тип	Описание
name	LABELS?	Официальное название места
alt_name	LABELS?	Альтернативное название места, которое чаще всего используется в устной речи посетителей
category	ENVIRONMENT-UNIT-CATEGORY	Категория, которая лучше всего описывает место
restriction	RESTRICTION-CATEGORY?	Категория, которая лучше всего описывает ограничение, применимое к месту
display_point	DISPLAY-POINT?	Координата используемая в качестве точечного представления места

Таблица 4: Виды категорий EnvironmentUnit (ENVIRONMENT-UNIT-CATEGORY)

Категория	Описание
road.main	Автомобильная дорога
road.dirt	Неасфальтированная автомобильная дорога
road.pedestrian.main	Основная пешеходная дорога (асфальтированный тротуар)
road.pedestrian.second	Второстепенная пешеходная дорога (например асфальтированные площади, служит для контрастного отделения от основной)
road.pedestrian.dirt	Неасфальтированная пешеходная дорога
grass	Область засаженная газоном
tree	Область высаженных деревьев (например аллеи в парке)
forest	Область природного леса

EnvironmentAnchor

Якорь представляет собой точку, используемую в качестве предпочтительного местоположения отображения определенного удобства ссылающегося на него. Связывает точку в пространстве с конкретным местом (Unit).

Таблица 5: Модель уличного якоря

Поле	Тип	Описание
unit_id	UUID	Идентификатор модели места (EnvironmentUnit)

EnvironmentAmenity

Уличное удобство моделирует физическое присутствие и местоположение объекта, который служит утилитарной цели или другому удобству. Аналогичен Amenity внутри помещений.

Таблица 6: Модель аннотации уличного удобства

Поле	Тип	Описание
name	LABELS?	Официальное название удобства
alt_name	LABELS?	Альтернативное название удобства, которое чаще всего используется в устной речи посетителей
category	ENVIRONMENT-AMENITY-CATEGORY	Категория удобства
detailLevel	NUMBER?	Уровень детализации, чем выше, тем важнее объект удобства
anchor_id	UUID	Идентификатор модели уличного якоря (EnvironmentAnchor)

Таблица 7: Виды категорий EnvironmentAmenity (ENVIRONMENT-AMENITY-CATEGORY)

Категория	Описание
parking.car	Автомобильная парковка
parking.bicycle	Велопарковка
banch	Скамейка
entrance.public	Открытый публичный пешеходный вход (калитка)
entrance.secure	Вход по пропускам

barrier	Контролируемое препятствие для проезда автомобильней (шлагбаум)
playground	Детская площадка

EnvironmentDetail

Детализация моделирует наличие, местоположение и вид физического объекта, распознавание которого в значительной степени зависит от пространственного контекста. Например дорожная разметка, перепад высот, ступеньки, заборы и т.д. Аналогичен Detail внутри помещений.

Таблица 8: Модель уличной детализации

Поле	Тип	Описание
category	ENVIRONMENT-DETAIL-CATEGORY	Категория детализации

Таблица 9: Виды категорий EnvironmentDetail (ENVIRONMENT-DETAIL-CATEGORY)

Категория	Описание
crosswalk	Разметка пешеходного перехода
road.marking.main	Дорожная разметка (например сплошная линия между полосами)
parking.marking	Разметка мест на парковке
parking.big	Жирная разметка мест на парковке (обычно используется для запрещения парковки в определённых местах)

fence.main	Препятствие пешеходному маршруту в явно физическом виде (забор)
fence.heigth	Неявное препятствие пешеходному маршруту (например резкий перепад высот)
stpdf	Ступеньки

Attraction

Точка интереса, один из видов аннотации, указывает на область сильного интереса посетителей, позволяет отображать на карте изображение, чтоб выделяться среди остальных аннотаций. Рекомендуется ассоциировать со зданиями имеющими внутреннюю планировку.

Таблица 10: Модель аннотации точки интересе

Поле	Тип	Описание
name	LABELS?	Официальное название точки интереса
alt_name	LABELS?	Альтернативное название точки интереса, которое чаще всего используется в устной речи посетителей
short_name	LABELS?	Короткое название менее чем из 2х букв для отображения внутри аннотации в случае отсутствия изображения
category	ATTRACTION-CATEGORY	Категория точки интереса
image	URL?	Ссылка на изображение отображаемое в аннотации
building_id	UUID?	Идентификатор модели здания (Building) ассоциированного с этой аннотацией

Таблица 11: Виды категорий Attraction (ATTRACTION-CATEGORY)

Категория	Описание
building	Аннотация здания
other	Любой другой вид аннотации

3.2.2. Поиск маршрута

Формат IMDF не содержит в себе информации необходимой для поиска маршрута, по этому я добавил описание односвязного графа, по которому можно будет построить маршрут. Для обратной совместимости с GeoJSON, граф описывается в формате GeoJSON и состоит из двух моделей:

Модели графа маршрута

PathNode

Описывает вершину графа. Геометрия обязана быть Point согласно GeoJSON. Для бесшовного построения маршрута, внутри одного графа, вершины могут находиться как на улице, так и в помещение, для верного отображения маршрута, каждая вершина обладает информацией о своём нахождение. Для вершин внутри зданий, указывается идентификатор модели здания (Building) и этажа (Level) на котором находится вершина.

Таблица 12: Модель вершины графа маршрутов

Поле	Тип	Описание
builing_id	UUID?	Идентификатор модели здания (Building) в котором находится вершина

level_id	UUID?	Идентификатор модели этажа (Level) в котором находится вершина
neighbours	UUID[]	Массив идентификатор соседних вершин в которые можно перейти
weights	NUMBER[]?	Массив коэффициентов весов для перехода к соседним вершинам, длинна массива обязана совпадать с длинной массива neighbours. Если массив не указан, используется массив из 1
category	PATH-NODE-CATEGORY	Категория вершины (используется для поиска маршрута с учётом параметров)
restriction	RESTRICTION-CATEGORY?	Категория описывающая применимое к вершине ограничение (используется для поиска маршрута с учётом параметров)

Таблица 13: Виды категорий PathNode (PATH-NODE-CATEGORY)

Категория	Описание
crosswalk	Разметка пешеходного перехода
stairs	Ступеньки
dirt	Неасфальтированная дорога
indoor.main	Маршрут внутри помещения

PathNodeAssociated

Перед поиском маршрута необходимо определить начальную и конечную вершину графа, для этого необходимо связать вершины графа с

отображаемыми пользователю аннотациями, для этой задачи служит модель PathNodeAssociated.

Таблица 14: Модель ассоциации вершин графа маршрутов с аннотациями

Поле	Тип	Описание
path_node_id	UUID	Идентификатор модели вершины маршрута (PathNode)
associeted_id	UUID	Идентификатор модели аннотации (Occupant/Amenity/Attraction/EnvironmentAmen ассоциированной с указанной вершиной

Если аннотация ассоциирована более чем с одной точкой маршрута (это может случиться с кабинетом у которого несколько входов), тогда система поиска маршрута рассмотрит все варианты и выберет кратчайший путь.

3.3. Конструктор расширенного стандарта

Несмотря на открытость формата и его мировую стандартизацию, открытого программного обеспечения для создания карт в этом формате нет, тем более нет программного обеспечения для моей расширенной версии формата, по этому я разработал свой собственный конструктор карт. Конструктор, на данный момент, не является конечным программным продуктом, и создан исключительно для решения утилитарной задачи – быстро и легко нарисовать карту. Конструктор написан на языке C# и использует возможности редактора Unity3D [**Unity3D**], а в частности редактор полигонов, иерархическую систему хранения объектов, компонентный подход к разработке, инспектор для редактирования параметров компонентов.

Было разработано дополнительное окно редактора для быстрого создания IMDF объектов, разработаны компоненты представляющие все основные

модели формата, написана система сериализации в IMDF. В конструкторе планировка представляется в виде иерархической структуры, что сильно ускоряет её создание. Например, комнаты Unit являются дочерними к этажу Level, которые дочерние к зданию Building. При сериализации, иерархическая структура нормализуется и все необходимые ссылки устанавливаются автоматически, кроме того по умолчанию комнаты Unit ”наследуют”адрес здания дописывая к нему своё название, что позволяет явно не указывать большую часть параметров формата. Для аннотаций создан отдельный компонент, которые объединяет в себе логику аннотации (Occupant) и якоря (Anchor), что значительно ускоряет процесс их размещения на карте.

Отдельно реализован инструмент для прокладывания графа маршрута с возможностью связать новую вершину с ближайшей или прошлой вершиной, кроме того, при добавлении ребра, по умолчанию инструмент делает его двунаправленным, то есть переход возможен в обе стороны.

[Скриншот конструктора IMDF формата](Картинка)

После сериализации получается полностью корректный формат IMDF расширенный моделями окружения. Обратная совместимость с форматом IMDF и базовым GeoJSON соблюдена.

3.4. Оцифровка карты

Процесс создания карты заключается в представление физических особенностей местности в форме многоугольников. Для увеличения продуктивности, конструктор позволяет привязать картинку к географическим координатам и отобразить на фоне редактора. В качестве первого приближения можно использовать снимки из космоса, однако они обладают рядом недостатков:

- низкое разрешение
- перспективные искажения, из-за которых высокие здания получаются сдвинутыми относительно фундамента
- кроны деревьев, которые полностью перекрывают пешеходные дорожки

3.4.1. Ортофотоплан

Для нивелирования недостатков снимков из космоса, я решил составить ортофотоплан кампуса университета. Это цифровое трансформированное изображение местности, созданное по перекрывающимся исходным фотоснимкам. Для фотоснимков использовался квадрокоптер DJI Mavic 2 Zoom. С помощью программы Copterus [**Copterus**] был составлен полётный план со следующими параметрами:

- высота полёта: 120м
- горизонтальное перекрытие: 85%
- вертикальное перекрытие: 70%

Было получено 1100 фотографий, которые впоследствии были обработаны программой WebODM [**WebODM**]. В результате получен ортофотоплан кампуса с разрешением 5см в каждом пикселе ()A.

[Картинка в приложение](Приложение)

ГЛАВА 4. РАЗРАБОТКА ПРИЛОЖЕНИЯ

Разработка приложение велась в среде Xcode. Был использован язык программирования Swift. В качестве архитектурного паттерна для разработки приложения был выбран Cacao MVC, это классический подход к разработке приложений для iOS, наиболее простой способ написания кода, который поддерживается абсолютным большинство библиотек, также приложения с таким паттерном проще и дешевле поддерживать, потому что есть много специалистов умеющих им пользоваться.

4.1. Пользовательский интерфейс

В разработке под iOS есть несколько вариантов верстки пользовательского интерфейса, важно перед началом разработки выбрать один из них и вести разработку в нём, так как переключиться по ходу разработки будет проблематично. Рассмотрим преимущества и недостатки каждого подхода

Storyboard

Базовый подход к созданию интерфейса, заключается в создание интерфейса с помощью специального конструктора в Xcode путём перетаскивания блоков на экран устройства. После запуска интерфейс конвертируется в UIKit объекты, к которым можно получить доступ с помощью специальных связок @IBOutlet

Достоинства:

- наглядность

Недостатки:

- сложность отслеживания изменений (файл интерфейса автосгенерированный xml код)
- сложность контроля через код
- сложно переиспользовать компоненты

Написание интерфейса кодом

Подход похожий на Storyboard, однако подразумевает ”ручное”создание UIKit объектов и размещение их друг относительно друга

Достоинства:

1. каждый элемент интерфейса является обычным объектом языка, что позволяет в любой момент получить к нему доступ
2. легко отслеживать изменения
3. полный контроль над интерфейсом
4. простота переиспользования компонентов

Недостатки:

- описание интерфейса занимает очень много кода
- отсутствует предпросмотр интерфейса, визуально можно оценить результат только после компиляции и запуска на устройстве

SwiftUI

Библиотека для написания интерфейса с помощью декларативного кода

Достоинства:

- малое количество кода даёт большой результат
- простота переиспользования компонентов
- предпросмотр интерфейса

Недостатки:

- на данный момент не полностью покрывает UIKit, и на решение многих примитивных задач потребуются много времени
- недостаточный контроль над интерфейсом

4.2. Структурные части пользовательского интерфейса

Пользовательский интерфейс моего приложения можно разделить на следующие части

- отображение карты
- отображение информации о выбранном объекте на карте

- вспомогательные всплывающие окна несвязанные с картой (например окно поделиться маршрутом)

В результате анализа, для разработки основной части интерфейса приложения был выбран способ написания его с помощью кода, однако для вспомогательных всплывающих окон, которые не имеют зависимостей в основной части интерфейса, используется SwiftUI. Такой подход позволил иметь полный контроль над картой и основной логикой приложения, однако оставил простоту написания для ”одностраничных”независимых вспомогательных окон

4.3. Чтение формата IMDF

Формат GeoJSON, который является базой IMDF, не подразумевает строгой типизации моделей, это делает десереализацию такого формата в строго типизированном языке нетривиальной задачей. Стандартный фреймворк для работы с картами MapKit поддерживает десереализацию идентификаторов (identifier) и геометрии (geometry) объектов GeoJSON в класс MKGeoJSONFeature, однако полезную нагрузку (properties) необходимо получать самостоятельно.

Листинг 4.1: Интерфейс объекта MKGeoJSONFeature

```
1 open class MKGeoJSONFeature : NSObject, MKGeoJSONObject {
2   open var identifier: String? { get }
3   open var properties: Data? { get }
4   open var geometry: [MKShape & MKGeoJSONObject] { get }
5 }
```

Для универсальной десереализации всех моделей был создан класс Feature шаблонизируемый классом Properties, конструктор класса сохраняет идентификатор и геометрию из MKGeoJSONFeature и декодирует Data в Properties

Листинг 4.2: Реализация класса Feature

```
1 class Feature<Properties: Decodable>: NSObject,
```

```
    IMDFDecodableFeature {
let identifier: UUID
let properties: Properties
let geometry: [MKShape & MKGeoJSONObject]

required init(feature: MKGeoJSONFeature) throws {
    identifier = UUID(uuidString: feature.identifier!)!
    geometry = feature.geometry

    if let propertiesData = feature.properties {
        properties = try JSONDecoder().decode(Properties.self, from
            : propertiesData)
      } else {
        throw IMDFError.invalidData
      }

    super.init()
  }
}
```

Модели IMDF необходимо наследовать от Feature передавая в качестве шаблона десереализуемую структуру Properties

Листинг 4.3: Пример объявления модели Unit

```
class Unit: Feature<Unit.Properties> {
    enum Category: String, Codable { ... }

    struct Properties: Codable {
        let name: LocalizedName?
        let alt\_name: LocalizedName?

        let level\_id: UUID
        let category: Category
        let restriction: Restriction?
        let display\_point: PointGeometry?
```

```
12      }
13  }
```

Для десереализации FeaturesCollection была написана функция decodeFeatures, с её помощью можно получить типизированный массив моделей

Листинг 4.4: Пример десереализации массива комнат Units

```
1  let imdfUnits = try decodeFeatures(Unit.self, path: unitPath)
2  static func decodeFeatures<T: IMDFDecodableFeature>(type: T.Type,
       path: URL) throws -> [T] {
3  let data = try Data(contentsOf: path)
4  let geoJSONFeatures = try MKGeoJSONDecoder().decode(data)
5
6  guard let features = geoJSONFeatures as? [MKGeoJSONFeature] else {
7      throw IMDFError.invalidType
8  }
9  let imdfFeatures = try features.map { try type.init(feature: $0) }
```

4.4. Отображение карты

Есть два способа отображения карты:

- использовать картографический фреймворк с возможностью добавления своих элементов наложения (оверлея). Есть несколько таких фреймворков:
- MapKit официальный фреймворк Apple, позволяет отображать векторную геометрию наложения, а так же аннотации. Поддерживает многопоточную отрисовку, а так же кеширования аннотаций для последующего переиспользования
- YandexMapKit фреймворк компании Яндекс позволяющий отображать Яндекс карты, добавлять на них аннотации. Сильно ограничен лицензионным соглашением и обладает плохой документацией
- MapBox iOS SDK предоставляет доступ к картам OpenStreetMap, добав-

ление наложения сильно ограничено. Для полноценного отображения карты необходимо загружать её на сервера MapBox

- отображать карту с помощью фрейморка низкоуровневой отрисовки графики Core Graphics. Данный способ наиболее универсален, но при таком подходе придётся полностью писать код для взаимодействия с вводом пользователя, придётся реализовывать жесты приближения и вращения карты, а так же оптимизировать производительность отображения наложений и аннотаций.

Был выбран оптимальный с точки зрения сложности реализации и качества результата фреймворк MapKit от компании Apple, несмотря на некоторые сложности в отображение наложений, он уже предоставляет готовые механизмы взаимодействия с картой. Кроме того, готовый фрейморк позволяет отображать наложения поверх обычной карты города благодаря чему, высоко детализированный кампус органично впишется в общую карту.

4.4.1. Отображение наложений на карту

После десереализации с помощью MKGeoJSONDecoder, геометрия представляется в виде массива композиции [MKShape & MKGeoJSONObject]

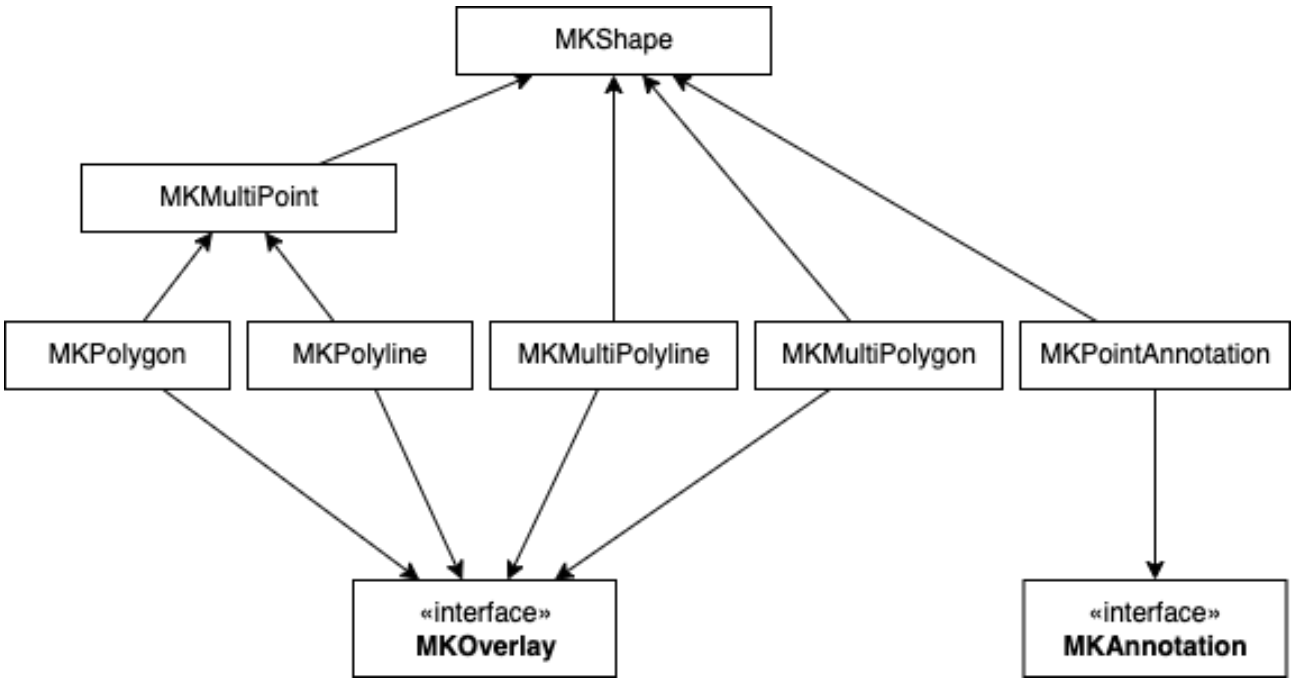

Рис. 4.1: Диаграмма классов GeoJSON.geometry

Объекты геометрии полученные после декодирования удовлетворяют интерфейсу MKOverlay. После добавление на экран MKMapView, с помощью функции *addOverlay(_ overlay: MKOverlay)* можно добавить геометрию для отображения на карте. После чего в делегате MKMapViewDelegate необходимо переопределить функцию *mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer*, которая для каждого объекта геометрии должна вернуть потокобезопасный экземпляр MKOverlayRenderer. Фреймворк предоставляет готовые реализации MKOverlayRenderer для всех видов геометрии. Во время разработки я столкнулся с проблемой, что векторное отображение возможно только при использовании готовых реализаций, из-за этого возникают серьёзные проблемы с кастомизанией, например для MKLineRenderer невозможно указать цвет обводки. Для базовой кастомизации я сделал интерфейс Styleble, который реализуют все виды наложения. Функция *configurate(_ renderer: MKOverlayRenderer)* позволяет делегировать объектам карты конфигурацию компонента отрисовки.

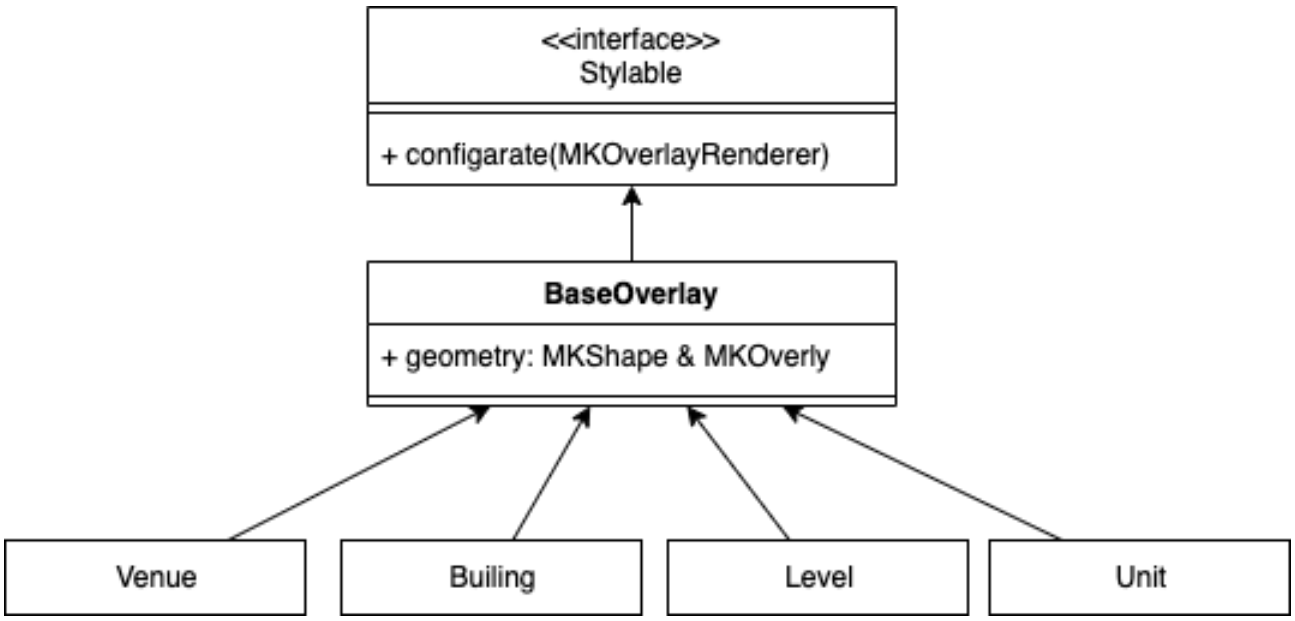

Рис. 4.2: Диаграмма классов объектов наложение

После чего, в функции делегата можно создать новый экземпляр MKOverlayRenderer, и сконфигурировать его приведя MKOverlay к интерфейсу Styleble

Листинг 4.5: Реализация функции создания и конфигурации компонентов MKOverlayRenderer

```
1  func renderer(for overlay: MKOverlay) -> MKOverlayRenderer {
2    switch overlay {
3        case is MKMultiPolygon: return MKMultiPolygonRenderer(overlay
             : overlay)
4        case is MKPolygon: return MKPolygonRenderer(overlay: overlay)
5        case is MKMultiPolyline: return MKMultiPolylineRenderer(
             overlay: overlay)
6        case is MKPolyline: return MKPolylineRenderer(overlay:
             overlay)
7        default: return MKOverlayRenderer(overlay: overlay)
8     }
9  }
10
11 func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay)
     -> MKOverlayRenderer {
```

```
12      let renderer = renderer(for: overlay)
13      (overlay as? Styleble)?.configurate(renderer: renderer)
14
15      return renderer
16  }
```

4.4.2. Отображение аннотаций на карте

Аннотации это текстовые сообщения, привязанные географическим координатам на карте. Добавляются на карту аналогично оверлею, фреймворк автоматически кеширует и переиспользует аннотации одного типа. Аннотации представляют собой UIView аналогичные тем, что используются для отображения всего остального интерфейса в UIKit. В моём приложении реализовано три типа аннотаций:

- OccupantAnnotation – используется для отображения аннотаций занимающих кабинеты, при выборе увеличивается и показывает пиктограмму категории аннотации. Например: учебный кабинет, лекторий, туалет
- AmenityAnnotation – используется для отображения удобств. Например: лестница, вендинговый аппарат, парковка
- AttractionAnnotation – используется для отображения точек интереса. Например: Главное здание

Все аннотации поддерживают четыре вида размеров: скрыт, маленький, нормальный, большой. Определение размера относительно приближения карты делегируются экземпляру объекта DetailLevelProcessor, благодаря этому один тип аннотации может обладать разным уровнем детализации в зависимости от представляемого вида объекта. Кроме того каждая аннотация может находиться в состояние ”выбрана”и в состояние ”закреплена”. Для переходов между всеми комбинациями состояний предусмотрены анимации.

4.5. Интернационализация

Так как после смены языка операционная система принудительно перезапускает приложение, механизма для смены языка во время работы приложения делать не нужно.

Локализация текстов

В XCode встроен готовый механизм локализации с помощью создания файлов Localizable.strings, который представляет собой структуру ”ключ = значение; чтобы получить локализованную строку, необходимо использовать let value = NSLocalizedString(key: ”key”) [**NSLocalizedString**]. Такой подход плох тем, что ключи передаются в строковом виде, и в случае отсутствия ключа в файле локализации случится ошибка времени выполнения. Для переноса обнаружения ошибки на время компиляции я использую библиотеку SwiftGen [**SwiftGen**] распространяемую по лицензии MIT [**MIT**], которая по файлу локализации генерирует статический класс, через который можно получить доступ только к тем значениям, которые точно есть в файле локализации. Пример использования: let value = L10n.key.

Локализация карты

Формат IMDF подразумевает хранение всех строковых представлений с помощью LABELS, которое представляет собой словарь [язык:строка]. Это значит что файл карты уже содержит все нужные локализации, и остаётся в момент десереализации выбрать нужную в зависимости от текущей локали приложения.

Локализация значений

В подробной информации о маршруте отображаются его расстояние и время на прохождение, в зависимости от локали пользователя должны от-

личаться единицы измерения расстояния, десятичный разделитель, локализация единиц измерения. Для решения этой задачи я использую встроенные в язык программирования MKDistanceFormatter [**MKDistanceFormatter**] и DateComponentsFormatter [**DateComponentsFormatter**] для форматирования дистанции и времени в строки с учётом текущей локали.

4.6. Адаптивные цвета

Для удовлетворения поставленным требованиям, в приложение было необходимо сделать поддержку переключения между тёмной и светлой темами. Это означает, что для каждого элемента интерфейса необходимо определить два варианта цвета, и переключаться между ними во время изменения системой темы оформления. В большинстве случаев для этого достаточно использовать стандартный компонент UIColor, а цвета создавать в формате ассетов XCode, это позволяет присвоить цветам названия, и для каждого цвета указать его варианты для светлой и тёмной темы, при изменении систменой темы, варианты цвета будут изменяться автоматически. Кроме того, для многих задач следует использовать стандартные системные цвета, доступные как статические поля класса UIColor. Например, для текста следует использовать UIColor.label [Полный список системных цветов](https://sarunw.com/posts/dark-color-cheat-sheet/) Используемая для локализации библиотеки SwiftGen, также генерирует класс с типизированным списком всех ассетов цветов, что позволяет решить проблему с получением ассета цвета по строковому ключу.

4.7. Оптимизация процесса разработки

В качестве системы контроля версий использовался GitHub, разработка велась согласно GitFlow. Для упрощения процесса тестирования и распространения тестовых версий использовался TestFlight. Был написан CI/CD использующий GitHub Actions, который на каждое изменение ветки dev, производил компиляцию проекта, подписывал её сертификатами и отправлял на сервера в TestFlight, после чего тестовая сборка ”по воздуху”устанавливалась на все

устройства тестировщиков. Для упрощения процесса компиляции и отправки была использована платформа Fastlane [**Fastlane**].

Стоит отметить, что репозиторий с приложением публичен, а для подписывания сборки необходимо использовать приватные сертификаты разработчика, которые в публичном репозитории не хранятся. Для решения этой проблемы был создан приватный репозиторий с сертификатами, которые скачиваются на CI сервер во время выполнения сборки, пароль для доступа к этому репозиторию хранится в специальном хранилище Github Secrets, значения из которого можно безопасно использовать в Action публичных репозиториев.

ГЛАВА 5. ФУНКЦИЯ ПОДЕЛИТЬСЯ

Одна из важных особенностей любого картографического приложения это возможность поделиться маршрутом. В моём приложении для этого предусмотрена возможность поделиться URL ссылкой на маршрут. После прокладывания маршрута, можно через меню ”поделиться”получить ссылку на маршрут. Кроме того к маршруту можно добавить приветственное сообщение, которое отобразится при открытии ссылки. Через это же меню можно создать красивый QR или AppClip коды.

5.1. Universal Links

Для того чтобы на устройствах iOS ссылка открывалась не в браузере, а в приложение, была интегрирована технологии Universal Links [**UniversalLinks**]. Для этого была включена поддержка ассоциации с доменами (Associated Domains) на портале разработчика, а так же в проекте Xcode в соответствующем разделе указан ассоциированный домен. После чего, при открытии ссылки, операционная система перенаправит пользователя на приложение, при этом передав ссылку в параметр userActivity у AppDelegate

Листинг 5.1: Получение ссылки из Universal Links

```
1 func application(\_ application: UIApplication, continue
    userActivity: NSUserActivity, restorationHandler: [Any?] -> Void
    ) -> Bool {
2   guard userActivity.activityType == NSUserActivityTypeBrowsingWeb,
        let url = userActivity.webpageURL else { return false }
3
4   print(url)
5   return true
6 }
```

5.2. AppClip

Наиболее распространённый случай использования ссылок на маршрут это размещение их в приглашениях на мероприятия, целевая аудитория таких приглашения – гости университета, у которых, вероятнее всего, приложение не будет установленно на устройтсво и Universal Links не сработает. Для таких случаев в моём приложении предусмотрена технология блиц-приложений AppClip [**AppClip**]. Блиц-приложение это отдельная версия программы, с помощью которой пользователь может получить быстрый доступ к определённым функциям приложения, при этом загрузка и установка полной версии из магазина приложений не требуется. Для блиц-приложение есть несколько вариантов запуска

- поднести устройство к NFC-метке
- отсканировать фирменную метку App Clip или стандартный QR-код
- перейти по URL ссылке в Safari, «Сообщениях» и других сервисах Apple
- запустить из баннера на сайте
- открыть предложенный App Clip в Apple Maps
- воспользоваться банером ”предложения Siri”на основе местоположения пользователя

После наступления одного из событий, операционная система отображает банер с предложением перейти в приложение, поле того как пользователь соглашается, мгновенно открывается упрощённая версия программы, в которую при наличии передаётся ссылка методом Universal Links.

[Картинка работы AppClip](Картинка)

В случае если пользователь выйдет из приложения, повторно открыть его можно будет через специальный ярлык, который будет существовать ещё несколько недель, после чего автоматически удалится.

5.3. Описание использования

Любой пользователь через установленное приложение может создать URL ссылку на маршрут и поместить её в QR или AppClip код, после чего, например, поместить в приглашение на мероприятие. Любой пользователь отсканировавший код, или перешедший по ссылке увидит этот маршрут на интерактивной карте. Если ссылка была открыта не на устройстве iOS, то откроется сайт с информацией о маршруте, иначе с использованием Universal Links будет открыта основная версия приложения, если она есть на устройстве или будет установлена и открыта блиц версия. После открытия будет проложен маршрут и отобразиться окно с информацией о нём.

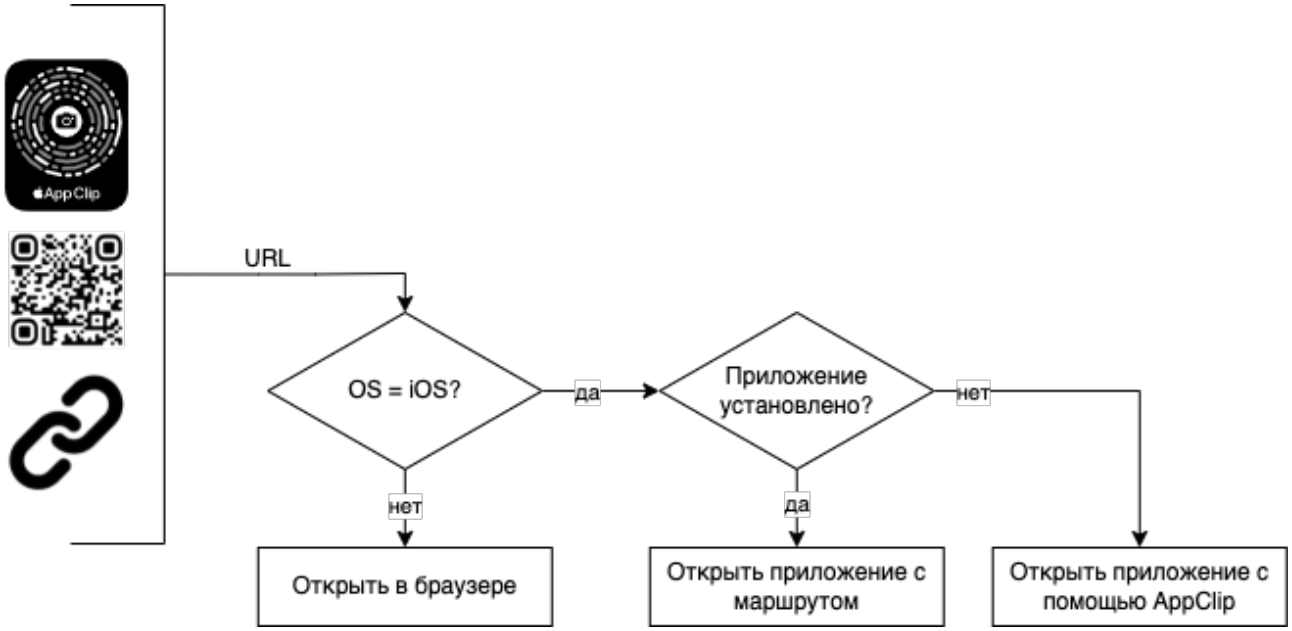

Рис. 5.1: Алгоритм открытия ссылки

ГЛАВА 6. СЕРВЕРНАЯ ЧАСТЬ

Для поддержки системы ”поделиться маршрутом” и ещё некоторых незначительных функций приложения необходим веб сервер. Для приложения был зарегистрирован домен polymap.ru и разработан веб сервер с использованием языка NodeJS [**NodeJS**] и фреймворка Express [**Express**]. Через Let’s Encrypt [**Let’s Encrypt**] были получены SSL сертификаты для поддержки протокола HTTPS.

Ассоциация домена с приложением

Для корректной работы Universal Links и AppClip необходимо ассоциировать приложение с доменом. Это означает, что на GET запрос по пути https://polymap.ru/.well-known/apple-app-site-association, должен вернуться текстовый файл, в котором должны быть записаны данные об ассоциации приложения в формате JSON. При этом сервер должен поддерживать протокол https. Пример файла:

Листинг 6.1: Файл ассоциации домена с приложением

```
{
  "applinks": {
    "details": [
      {
        "appIDs": [ "ABCDE12345.com.example.app" ],
        "paths": ["/"]
      }
    ]
  }
}
```

Для поддержки AppClip в вышеуказанный файл необходимо добавить соответствующий раздел:

Листинг 6.2: Ассоциация домена с AppClip

```
{
  ...
  "appclips": {
    "apps": ["ABCED12345.com.example.MyApp.Clip"]
  }
}
```

В момент открытия ссылки пользователем и наличие приложения ассоциированного с этой ссылкой, операционная система обращается к серверам Apple, которые обращаются к корневому домену ссылки и проверяют наличие ассоциации с приложением. Эта проверка кешируется на серверах Apple на 24 часа, что полностью снимает нагрузку с пользовательских серверов.

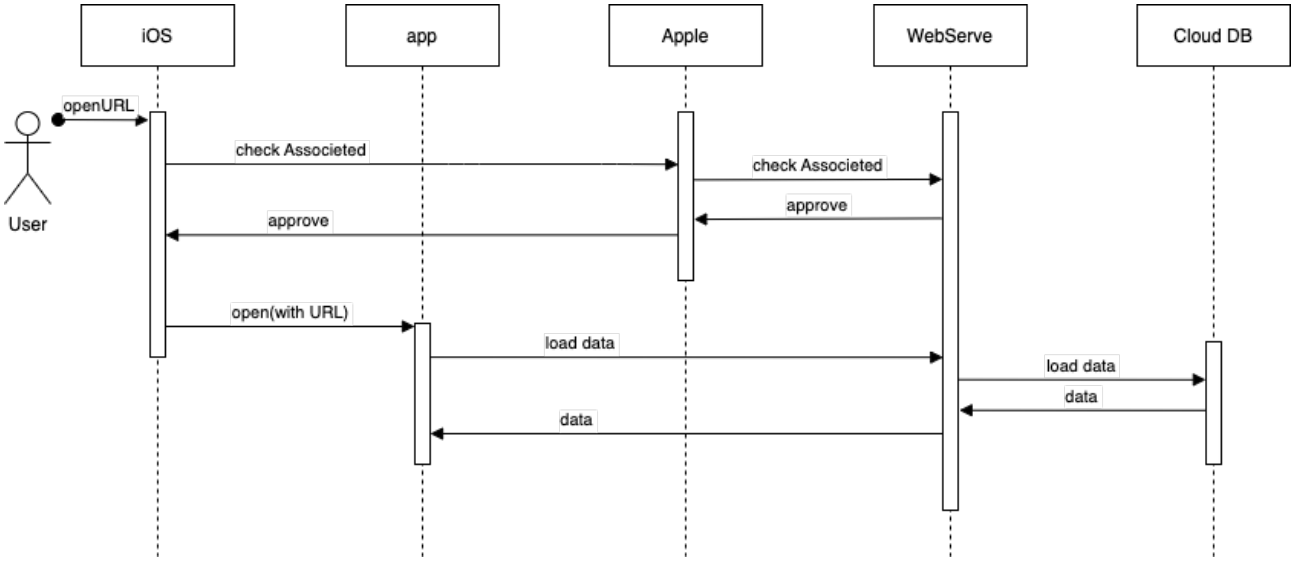

Рис. 6.1: Диаграмма взаимодействия при открытии ссылки

Поддержка функции поделиться

Для того чтоб поделиться маршрутом необходимо передать следующие параметры:

- UUID начала маршруту
- UUID завершения маршрута
- Параметры построения маршрута

- Разрешение менять параметры
- Приоритет асфальтированным маршрутам
- Разрешение служебных проходов
- Приветственное сообщение

Технически все эти параметры можно поместить в часть qurey параметров URL, однако тогда ссылка получится длинной и отпугивающей пользователя, кроме того, QR код для длинной ссылки потребует высокой детализации, что означает более долгое сканирование и менее привлекательный вид. AppClip код и вовсе позволяет закодировать около 50 символов, некоторые сочетания которых кодируются оптимальнее, однако длинную ссылку с несколькими UUID закодировать точно не получится. ![Пример QR кода для ссылок разной длины]() ![20 символов]() ![80 символов]()

Для сокращения ссылки была разработана следующая система: пользователь POST запросом отправляет на сервер все параметры связанные с маршрутом, они сохраняются в базу данных, а пользователю отдаётся ID этой записи. После чего пользователь может сделать запрос на сервер с указанным ID и получить сохранённые параметры маршрута.

Генерация AppClip кодов

Сложность генерации AppClip кодов состоит в том, что это закрытый формат представления данных и получить этот код можно только с помощью специальной CLI программы [**AppClipGenerator**], существующей только для системы MacOS. Из-за этого сгенерировать AppClip код прямо на устройстве не возможно, по этому генерация кода была вынесена на сервер. GET запрос на адрес polymap.ru/api/appclip-code генерирует и возвращает AppClip код, принимая следуюшие query параметры:

Таблица 15: Поддерживаемые query параметры генератора AppClip кодов

Параметр	Пример	Описание

id	5	ID кода
background	fff	Цвет фона в формате HEX
primary	000	Основной цвет в формате HEX
secondary	888	Второстепенный цвет в формате HEX
badgeTextColor	888	Цвет текста в формате HEX
logo	camera	Ваиант логотипа в центре кода. [camera, nfc]
useBadge	true	Разрешить использовать рамку вокург кода
type	888	Тип файла [svg, png]
width	512	В случе png изображения его ширина в пик-силях. Максимум 2048

С помощью этих параметров можно полностью кастомизировать код и получить любой из возможных его вариантов.

Так как сервер запускается на операционной системе Linux, а программа для генерации существует только для MacOS, смысловая нагрузка когда, прерывистые окружности, в формате SVG предгенерирована на MacOS для каждого варианта ID, сервер берёт нужный файл и оформляет код согласно пришедшем параметрам. Данное решение уникально и не встречается ни в одном другом приложении.

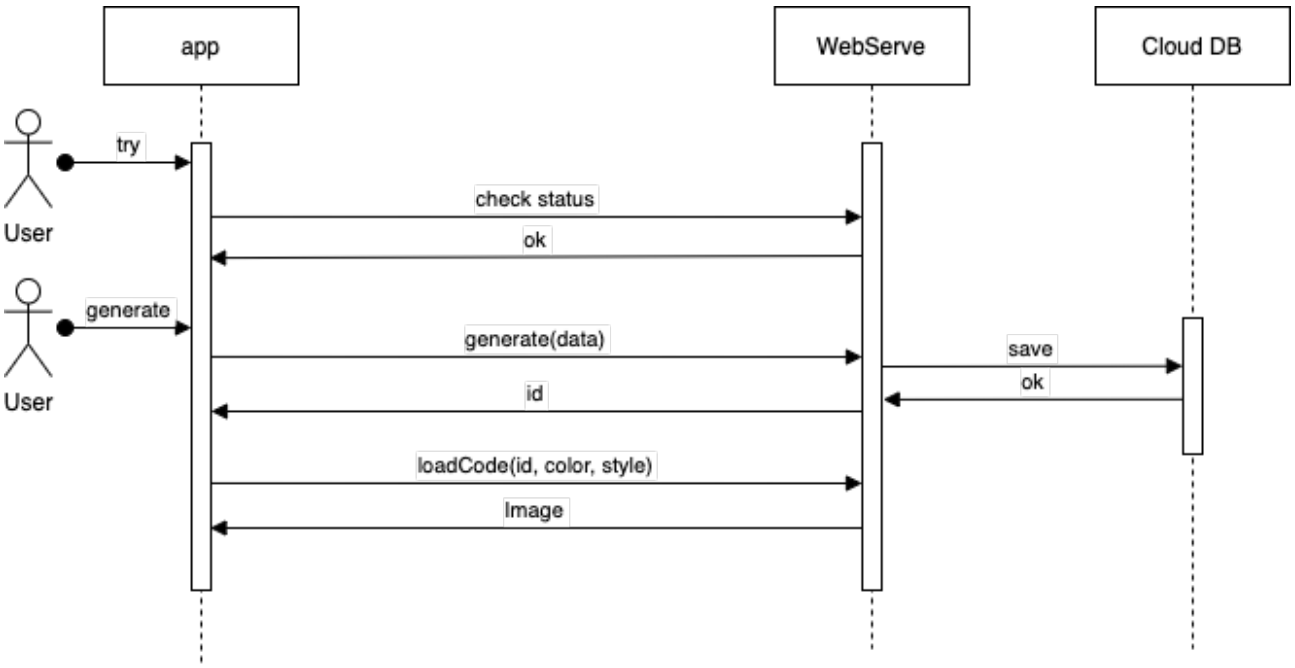

Рис. 6.2: Схема работы диалога поделиться

ЗАКЛЮЧЕНИЕ

В ходе работы был проведён анализ существующих на рынке решений, рассмотрены их положительные и отрицательные стороны. На основе положительных сторон были составлены требования к разработке. Была разработана методика картографии на основе формата IMDF. Формат был расширен, был разработан редактор для расширенного формата, составлена детализированная карта кампуса СПбПУ.

В результате работы было разработано мобильное приложения для операционной системы iOS удовлетворяющее всем стандартам качественной современной разработки, в том числе поддержка всех видов актуальных устройств, светлой и тёмной темы, интернационализации.

Приложение позволяет пользователю просматривать карту используя для этого привычные жесты управления, осуществлять поиск по аннотациям и строить маршруты бесшовно переходящие с улицы в помещения.

После построения маршрута пользователь может создать красиво оформленный QR или AppClip код с заложенным маршрутом.

6.1. Планы на развитие

В дальнейшем планируется выпустить унифицированную версию приложения, с возможностью скачивать карты из интернета и после этого отображать их. Для неё будет необходимо разработать веб версию конструктора карт, чтобы каждый желающий смог построить план интересующего его заведения и открыть его в универсальном приложении. Кроме того, я планирую выпустить Android версию приложения.

СПИСОК ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А. ПРИЛ

Текст приложения

ПРИЛОЖЕНИЕ Б. ПРИЛ

Текст приложения

ПРИЛОЖЕНИЕ В. ПРИЛ

Текст приложения