Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики
Направление подготовки Информационные системы и технологии
Кафедра информационных систем и технологий

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
Разработка мобильного веб-сервиса для обмена данными о местоположении пользователей

УДК 004.75:004.455:316.472

Студент

Группа	ФИО	Подпись	Дата
8ИМ5А	Галузо Кирилл Борисович		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Шестаков Николай Александрович	к.т.н.		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Попова Светлана Николаевна	к.т.н		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
ассистент	Акулов Петр Анатольевич			

ДОПУСТИТЬ К ЗАЩИТЕ:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
ИСТ	Мальчуков Андрей Николаевич	к.т.н		

Томск – 2017 г.

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики
Направление подготовки Информационные системы и технологии
Кафедра Вычислительной техники

УТВЕРЖДАЮ:
Зав. кафедрой

_____ _____ _____
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

магистерской диссертиции

Студенту:

Группа	ФИО
8ИМ5А	Галузо Кириллу Борисовичу

Тема работы:

Разработка мобильного веб-сервиса для обмена данными о местоположении пользователей	
Утверждена приказом директора Института кибернетики (дата, номер)	

Срок сдачи студентом выполненной работы: (дата)	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Заказ на разработку мобильного веб-сервиса
Перечень подлежащих исследованию, проектированию и разработке вопросов	- Постановка задачи разработки - Проектирование архитектуры, разрабатываемого приложения - Разработка приложения - Заключение по работе
Введение	
1 Обзор литературы	
2 Проектирование	
3 Реализация	
4 Результаты реализации	
5 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	
6 Социальная ответственность	
Заключение	
Список публикаций	

Список используемых источников
Приложения
Консультанты по разделам выпускной квалификационной работы

Раздел	Консультант
«Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»	Попова Светлана Николавена
«Социальная ответственность»	Акулов Петр Анатольевич

Названия разделов, которые должны быть написаны на русском и иностранном языках:
Проектирование, Реализация

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Шестаков Н.А.	к.т.н., доцент		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ5А	Галузо Кирилл Борисович		

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ООП

Код результатов	Результат обучения (выпускник должен быть готов)
Общепрофессиональные компетенции	
Р1	Воспринимать и самостоятельно приобретать, развивать и применять математические, естественнонаучные, социально-экономические и профессиональные знания для решения нестандартных задач, в том числе в новой или незнакомой среде и в междисциплинарном контексте.
Р2	Владеть и применять методы и средства получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе в глобальных компьютерных сетях.
Р3	Демонстрировать культуру мышления, способность выстраивать логику рассуждений и высказываний, основанных на интерпретации данных, интегрированных из разных областей науки и техники, выносить суждения на основании неполных данных, анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями.
Р4	Анализировать и оценивать уровни своих компетенций в сочетании со способностью и готовностью к саморегулированию дальнейшего образования и профессиональной мобильности. Владеть, по крайней мере, одним из иностранных языков на уровне социального и профессионального общения, применять специальную лексику и профессиональную терминологию языка.
Профессиональные компетенции	
Р5	Разрабатывать стратегии и цели проектирования, критерии эффективности и ограничения применимости, новые методы, средства и технологии проектирования геоинформационных систем (ГИС) или промышленного программного обеспечения.
Р6	Планировать и проводить теоретические и экспериментальные исследования в области создания интеллектуальных ГИС и ГИС технологии или промышленного программного обеспечения с использованием методов системной инженерии.
Р7	Осуществлять авторское сопровождение процессов проектирования, внедрения и сопровождения ГИС и ГИС технологий или промышленного программного обеспечения с использованием методов и средств системной инженерии, осуществлять подготовку и обучение персонала.
Р8	Формировать новые конкурентоспособные идеи в области теории и практики ГИС и ГИС технологий или системной инженерии программного обеспечения. Разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач. Организовывать взаимодействие коллективов, принимать управленческие решения, находить компромисс между различными требованиями как при долгосрочном, так и при краткосрочным планировании.
Общекультурные компетенции	
Р9	Использовать на практике умения и навыки в организации исследовательских, проектных работ и профессиональной эксплуатации современного оборудования и приборов, в управлении коллективом.
Р10	Свободно пользоваться русским и иностранным языками как средством делового общения.
Р11	Совершенствовать и развивать свой интеллектуальный и общекультурный уровень. Проявлять инициативу, в том числе в ситуациях риска, брать на себя всю полноту ответственности.
Р12	Демонстрировать способность к самостоятельному обучению новым методам исследования, к изменению научного и научно-производственного профиля своей профессиональной деятельности, способность самостоятельно приобретать с помощью информационных технологий и использовать в практической деятельности новые знания и умения, в том числе в новых областях знаний, непосредственно не связанных со сферой деятельности, способность к педагогической деятельности.

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики
Направление подготовки 09.04.02 Информационные системы и технологии
Кафедра информационных систем и технологий
Период выполнения весенний семестр 2016/2017 учебного года)

Форма представления работы:

Магистерская диссертация

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
3.02.2017 г	Анализ предметной области, сбор исходных данных	5
10.02.2017 г	Проектирование архитектуры приложения	10
16.02.2017 г	Изучение методов взаимодействия клиент-сервер	5
20.02.2017 г	Изучение методов определения местоположения на устройствах, работающих на платформе iOS	5
25.02.2017	Реализация основного функционала	45
2.05.2017 г	Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	10
12.05.2017 г	Социальная ответственность	10
18.05.2017 г	Обязательное приложение на иностранном языке	5
30.05.2017 г	Оформление пояснительной записки	5

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент каф. ИСТ	Шестаков Н.А.	к.т.н.		

СОГЛАСОВАНО:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
ИСТ	Мальчуков А.Н.	к.т.н		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8ИМ5А	Галузо Кирилл Борисович

Институт		**Кафедра**	
Уровень образования	Магистр	**Направление/специальность**	09.04.02 Информационные системы и технологии

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих*	
2. *Нормы и нормативы расходования ресурсов*	
3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого и инновационного потенциала НТИ*	
2. *Разработка устава научно-технического проекта*	*Планирование структуры НТИ, определение трудоемкости работ и составление календарного графика распределения работ*
3. *Планирование процесса управления НТИ: структура и график проведения, бюджет, риски и организация закупок*	
4. *Определение ресурсной, финансовой, экономической эффективности*	*Определение научно-технического уровня проекта*
Перечень графического материала *(с точным указанием обязательных чертежей):*	
1. *График проведения и бюджет НТИ* 2. *Оценка ресурсной, финансовой и экономической эффективности НТИ*	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель	Хаперская Алена Васильевна			

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ5А	Галузо Кирилл Борисович		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8ИМ5А	Галузо Кирилл Борисович

Институт		**Кафедра**	
Уровень образования	Магистр	**Направление/специальность**	09.04.02 Информационные системы и технологии

Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения	Объектом исследования являются среда разработки XCode, язык программирования Swift; использование эмуляторов мобильных устройств в среде XCode; Целью магистерской диссертации (предметом) является разработка мобильного веб-сервиса для обмена данными о местоположении пользователей..
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1 Производственная безопасность 1.1 Анализ выявленных вредных факторов при разработке и эксплуатации проектируемого решения в следующей последовательности. 1.2 Анализ выявленных опасных факторов при разработке и эксплуатации проектируемого решения в следующей последовательности.	Производственная безопасность на стадии разработки веб-системы. 1.1 В качестве вредных факторов выделены: – Недостаточная освещенность рабочей зоны; – Умственное перенапряжение; – Монотонный режим работы. 1.2. В качестве опасных факторов выделены: – Опасность поражения электрическим током; – Опасность возникновения пожара.
2 Экологическая безопасность: 2.1 Анализ воздействия объекта на окружающую среду; 2.2 Разработать решения по обеспечению экологической безопасности со ссылками на НТД по охране окружающей среды.	2.1 Влияние объекта исследования на окружающую среду: – Образование мусора. – Утилизация компьютерной техники. 2.2 Мероприятия по защите окружающей среды.
3 Безопасность в чрезвычайных ситуациях: 3.1 Перечень возможных ЧС при разработке и эксплуатации проектируемого решения; 3.2 Выбор наиболее типичной ЧС; 3.3 Разработка действий в результате возникшей ЧС и мер по ликвидации её последствий.	3.1 Возможные чрезвычайные ситуации: – Пожар; – Поражение электрическим током; 3.2 Типичная чрезвычайная ситуация: – Возгорание (пожар); 3.3 Мероприятия по предотвращению наиболее типичной ЧС – пожара, согласно нормативным документам: НПБ 105-03 и ППБ 01–03.

4 Правовые и организационные вопросы обеспечения безопасности: 4.1 Специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; 4.2 Организационные мероприятия при компоновке рабочей зоны.	4.1 Описание правовых норм для работ, связанных с работой за ПЭВМ согласно следующим документам: – Трудовой кодекс Российской Федерации" от 30.12.2001 N 197-ФЗ (ред. от 30.12.2015); 4.2 Влияние реализации разрабатываемой системы на конечных пользователей

Дата выдачи задания для раздела по линейному графику	22.02.2017

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Акулов Петр Анатольевич			

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ5А	Галузо Кирилл Борисович		

РЕФЕРАТ

Выпускная квалификационная работа содержит 115 страниц, 47 рисунков, 17 таблиц, 27 источников, 1 приложения.

Ключевые слова: мобильное приложение, iOS, геолокация.

Цель работы: создание мобильного сервиса, предназначенного для обмена данными о местоположении с другими пользователями. Мобильное приложение должно быть разработано для использования на устройствах, работающих на операционной системе iOS. Мобильное приложение должно взаимодействовать с сервером для обеспечения обмена данными о местоположении, профиле между пользователями. Точность определения местоположения не должна уступать аналогам. Разрабатываемое приложение должно иметь базовый инструментарий, такой как отображение пользователей на карте, создание и редактирование профиля, создание и редактирование групп пользователей. Процесс регистрации в приложении должен быть быстрым и простым.

Изложены результаты анализа конкурентов, литературы по исследуемой проблеме, алгоритмы решения поставленной задачи.

Разработано мобильное приложение, предназначенное для определения и визуализации местоположения родных и близких людей. В приложении предусмотрена процедура регистрации, создания и заполнения профиля, создания и редактирования групп пользователей, визуализации данных на карте.

Приложение находится в свободном доступе в интернет-магазине AppStore и имеет возрастное ограничение 4+.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

iOS – мобильная операционная система, разработанная корпорацией Apple.

AppStore – интернет-магазин, реализующий цифровую продукцию, предназначенную для устройств, работающих на операционных системах от корпорации Apple.

UIKit – фреймворк, разработанный корпорацией Apple, предоставляющий инфраструктуру для создания и управления приложений, разрабатываемых на платформы iOS и tvOS.

MVC (Model View Controller) – архитектурный паттерн.

MVP (Model View Presenter) – архитектурный паттерн.

MVVM (Model ViewModel Model) – архитектурный паттерн, используемый в разработке мобильных приложений на платформе iOS.

VIPER – архитектурный паттерн, используемый в разработке мобильных приложений на платформе iOS.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 13
1. ОБЗОР ТЕХНОЛОГИЙ 15
1.1. Геолокация 15
1.2. Архитектурные паттерны iOS-приложений 15
1.2.1. MVC 16
1.2.2. MVP 17
1.2.3. MVVM 18
1.2.4. VIPER 19
1.3. Методы взаимодействия клиент-сервер 22
1.3.1. HTTP-запросы 22
1.3.2. Формат JSON 23
1.3.3. Взаимодействие через сокеты 24
2. ТРЕБОВАНИЯ К СИСТЕМЕ 26
2.1. Варианты использования 26
2.1.1. ВИ состояний приложения 26
2.1.2. ВИ «Вход»/«Регистрация» 27
2.1.3. ВИ «Главный экран» 29
2.1.4. ВИ «Поиск» 30
2.1.5. ВИ «Карта» 31
2.1.6. ВИ «Группы» 33
2.1.7. ВИ «Создать группу» 35
2.1.8. ВИ «Настройки» 36
2.1.9. ВИ «Профиль» 37
3. РЕАЛИЗАЦИЯ 41
3.1. Архитектура 41
3.2. Библиотеки для взаимодействия с сервером 45
3.2.1. OMGHTTPURL 46
3.2.2. PromiseKit 46
3.2.3. Soket.IO 48
3.2.4. ObjectMapper 49
4. РЕЗУЛЬТАТ 52
5. РАЗДЕЛ «ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ» 60

6. РАЗДЕЛ «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ» 72
ЗАКЛЮЧЕНИЕ 89
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 90
ПРИЛОЖЕНИЕ А 94

ВВЕДЕНИЕ

В настоящее время мобильные устройства являются неотъемлемой частью быта почти каждого человека.

Рынок мобильных приложений растет с каждым годом, а значит появляется спрос на различные мобильные приложения, которые позволят автоматизировать и упростить повседневные потребности пользователей.

Существует множество мобильных приложений, использующих геолокационные сервисы, и широта применения этих геолокационных сервисов весьма велика.

В частности, они позволяют в реальном времени отслеживать местоположение других пользователей. Например, такие приложения могут быть использованы для оповещения и наблюдения за перемещениями детей их родителями, для организации работы курьерских служб, такси и т.д.

Целью работы является создание мобильного сервиса, предназначенного для обмена данными о местоположении с другими пользователями. Мобильное приложение должно быть разработано для использования на устройствах, работающих на операционной системе iOS. Мобильное приложение должно взаимодействовать с сервером для обеспечения обмена данными о местоположении, профиле между пользователями. Точность определения местоположения не должна уступать аналогам. Разрабатываемое приложение должно иметь базовый инструментарий, такой как отображение пользователей на карте, создание и редактирование профиля, создание и редактирование групп пользователей. Процесс регистрации в приложении должен быть быстрым и простым.

В разработанном мобильном приложении пользователю предоставляется возможность сформировать группы пользователей по категориям и отслеживать местоположение этих пользователей, создать и редактировать свой профиль.

В первой главе проведен обзор технологий. Изучены различные шаблоны проектирования архитектуры мобильных приложений, методы взаимодействия клиент-сервер.

Во второй главе рассмотрены варианты использования и приведены описания каждого варианта.

В третьей главе описана выбранная архитектура, выбранные фреймворки для клиент-серверного взаимодействия, приведены примеры реализации некоторых частей приложения.

В четвертой главе приведены эскизы интерфейса.

В пятой главе оценен коммерческий потенциал и перспективность разработанной программы с позиции ресурсоэффективности и ресурсосбережения. Определена ресурсная, финансовая, бюджетная, социальная и экономическая эффективность разработки.

В шестой главе проведен анализ выявленных опасных и вредных факторов проектируемой производственной среды, воздействия на атмосферу, гидросферу и литосферу. Приведены правовые и организационные вопросы обеспечения безопасности.

1. ОБЗОР ТЕХНОЛОГИЙ

1.1. Геолокация

Геолокация – определение географического положения объектов, таких как радары, мобильные устройства, компьютеры, подключенные к сети Интернет и т.д. Для геолокации зачастую используются методы радионавигации, а так же геоинформационные системы. Так же для геолокации могут быть использованы базы данных станций мобильной связи, в случаях, когда GPS-сигнал недоступен, однако такой способ обладает меньшей точностью, нежели в GPS. Геолокация интернет-устройств может быть произведена посредством связывания с реальным местоположением IP-адреса, MAC-адреса или Wi-Fi позиционирования.

Использование геолокации позволяет быстро определить свое местоположение и с ориентироваться на местности, проложить маршрут до точки назначения. Так же легко найти организацию, учереждение, кафе рестораны и т.д. в незнакомом городе.

Геолокация широко используется логистических кампаниях, компаниях, деятельность которых связна с перевозкой грузов и пассажиров.

Геолокация является мощным инструментов для рекламной сферы. Она позволяет использовать гео-таргетированную рекламу.

1.2. Архитектурные паттерны iOS-приложений

Архитектурные паттерны – переиспользуемые решения общих проблем в разработке мобильных приложений. Это шаблоны, которые помогают с легкостью сопровождать проект и делает его более понятным для других разработчиков. Существует ряд архитектурных решений при разработке приложений на iOS-платформе, каждый из которых имеет свои преимущества и недостатки. Признаками хорошей архитектуры являются:

- сбалансированное распределение обязанностей между сущностями с жесткими ролями;
- тестируемость;
- простота использования и низкая стоимость обслуживания.

Распределение обязанностей осуществляется по принципу единой ответственности, который гласит, что каждый объект должен иметь одну ответственность, которая должна быть инкапсулирована.

1.2.1. MVC

MVC (Model-View-Controller) – паттерн, разделяющий приложение на 3 части – модель (Model), представление (View) и контроллер (Controller) [16].

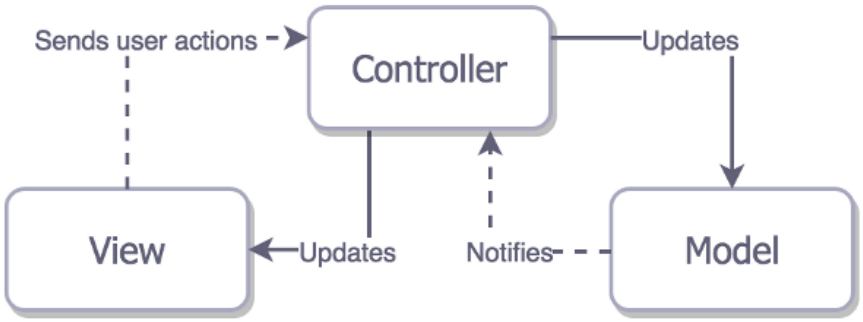

Рисунок 1.2.1.1 – схема традиционного MVC [16]

В традиционном MVC представление не хранит состояния в себе, а контроллер изменяет представление при изменении модели. Контроллер является посредником между моделью и представлением, то есть модель и представления друг о друге не знают ничего, другими словами они не связаны напрямую между собой [27].

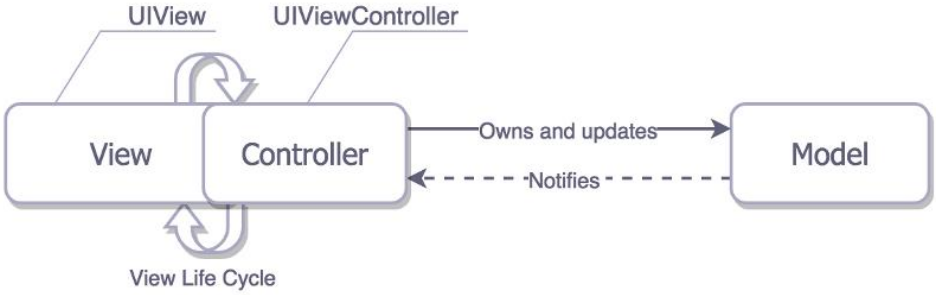

Рисунок 1.2.1.2 – схема Cocoa MVC от Apple [16]

В MVC от Apple контроллер вовлечен в жизненный цикл представления, следовательно не является отдельной сущностью. Представление и контроллер находятся в одном файле. Вся ответственность представления заключается в передаче действий контроллеру, но на самом деле ViewController является делегатом и источником данных, местом запуска и отмены запросов на сервер и т.д. Проблема особенно проявляется при написании юнит-тестов. Так как эти

тесты предназначены для проверки на корректность работы отдельных модулей приложения, то при массивном представлении-контроллере, написание таких тестов становится затруднительным. В таком случае необходимо заменять представление мок-объектами и имитировать их жизненный цикл.

Если оценить Cocoa MVC с точки зрения признаков хорошей архитектуры, то получается [16]:

- распределение: представление и модель разделены, но представление и контроллер тесно связаны;
- тестируемость: из-за тесной связности представления и контроллера и плохого распределения тестируется только модель;
- простота использования: по сравнению с другими паттернами имеет меньшее количество код и является более понятным для неопытных разработчиков.

1.2.2. MVP

MVP (Model View Presenter) разделяет приложение на 3 части – модель (Model), представление (View) и презентер (Presenter).

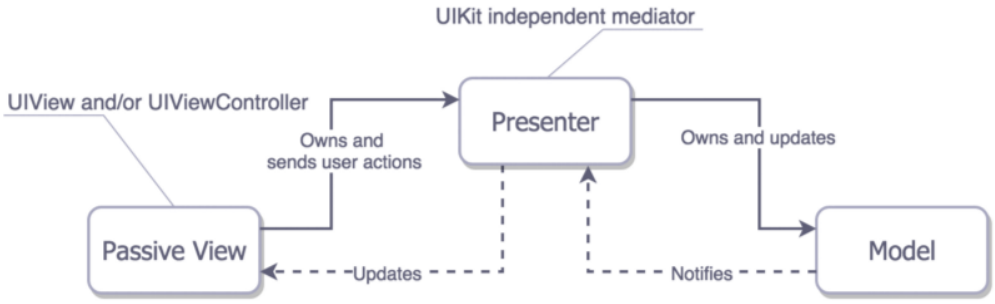

Рисунок 1.2.2 – схема MVP [16]

Данный паттерн подразумевает, что представление пассивно, а презентер не имеет отношения к жизненному циклу представления. Последний управляет обновлением представления в соответствии с состоянием и новыми данными, в нем нет кода разметки. При тестировании объекты представления можно легко заменить мок-объектами.

Однако в MVP проявляется проблема сборки проекта, которая возникает в следствии наличия трех действительно отдельных слоев. Так как представление и модель не должны знать друг о друге, выполнять сборку в презентуемом

представлении – неправильно, а значит нужно создать некий сервис – роутер (Router), который будет осуществлять сборку и презентацию представление-представление.

MVP с точки зрения признаков хорошей архитектуры:

- распределение: ответственность, по большей части, разделена между презентером и моделью в то время как представление ничего не делает, т.к. оно пассивно;
- тестируемость: в следствии пассивности представления появляется возможность проверить большую часть бизнес-логики приложения, что говорит о хорошей тестируемости;
- простота использования: программного кода больше чем в MVC, наряду с простотой идеи.

1.2.3. MVVM

MVVM (Model View ViewModel) – паттерн так же, как и другие паттерны из серии MV(X) делит приложение на 3 части – представление, модель и представление модели (ViewModel) в качестве посредника.

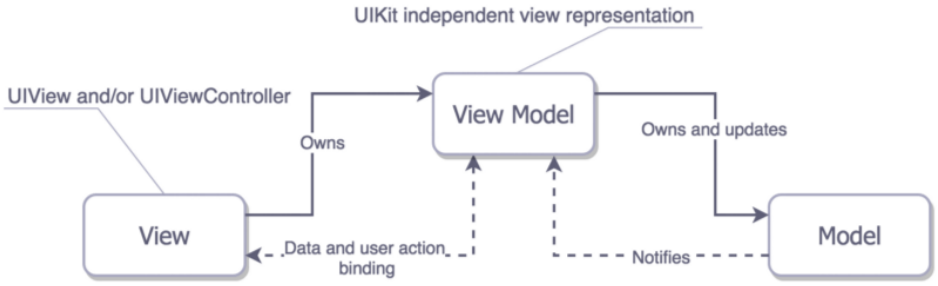

Рисунок 1.2.3 – схема MVVM [16]

MVVM очень схож с MVP тем, что рассматривает ViewController как представление, и в нем нет непосредственной связи между представлением и моделью.

В данном паттерне связываются не представление и модель, а представление и представление модели. Такие связи называются биндингами. Представление модели в среде iOS – независимое от UIKit представление представления (View) и ее состояния.

Представление модели вызывает изменение модели и самостоятельно обновляется с изменений моделью. Так как связь происходит между представлением и представлению модели, то представление так же обновится.

Использование полноразмерных фреймворков для функционального реактивного программирования (ReactiveCocoa, RxSwift, PromiseKit) позволяет реализовывать все преимущества MVVM-паттерна.

MVVM с точки зрения признаков хорошей архитектуры:

- распределение: представление имеет больше обязанностей, по сравнению с представлением из MVP-паттерна. Это обусловлено тем, что представление из MVVM обновляет свое состояние посредством связываний, в то время как представление из MVP направляет все события в презентер, в обязанности которого входит обновление представления;
- тестируемость: представление модели ничего не знает о представлении, что дает возможность легко протестировать ее, представление так же можно протестировать, но для этого потребуются не написание юнит-тестов, а создание тестов графического интерфейса;
- полнота использования: объем кода сопоставим с объемами кода в MVP-паттерне, но за счет связываний нет необходимости вручную направлять все события их представления в презентер и обновлять это представление.

MVVM-паттерн сочетает в себе все преимущества паттернов серии MV(X). В нем не требуется дополнительный код для обновления представления, в результате применения биндингов на стороне представления, При всем этом, тестируемость находится на хорошем уровне.

1.2.4. VIPER

VIPER (View Interactor Presenter Entity Router) – архитектурный паттерн, который разделяет приложение на 5 частей.

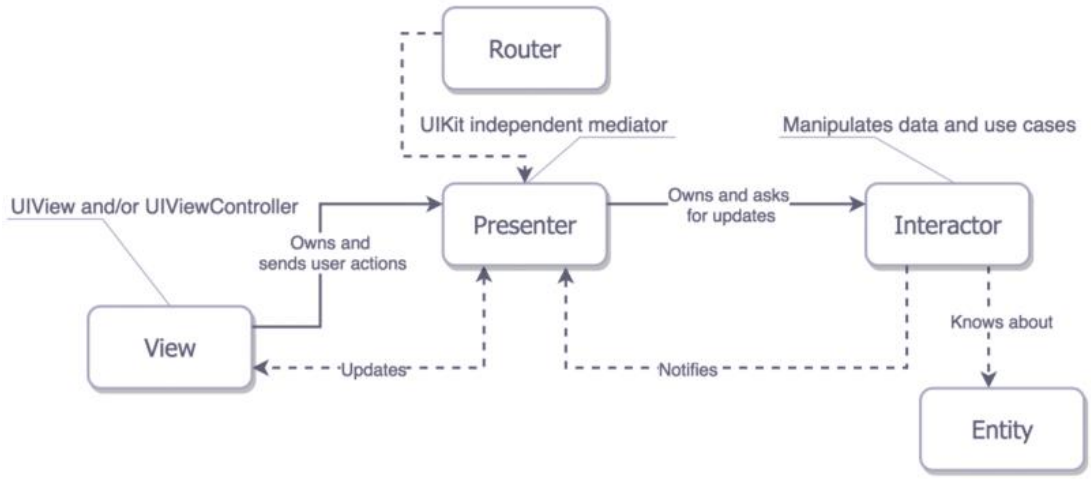

Рисунок 1.2.4 – схема VIPER[16]

Интерактор (Interactor) – слой, содержащий бизнес-логику приложения, которая связана с данными (Entities). Как пример, создание новых экземпляров сущностей, отправку или получение таковых с сервера. В этих целях используются различные сервисы и менеджеры, которые не являются частью модуля VIPER, а выступают как внешние зависимости. Задачи в интеракторе выполняются независимо от пользовательского интерфейса.

Презентер (Presenter) – слой, который содержит в себе бизнес-логику зависимую от интерфейса. Презентер собирает выходные данные от взаимодействия с пользователем и может передать их интерактору или роутеру. Так же презентер получает входные данные от интерактора или роутера и преобразовывает их в состояние, наиболее эффективное для отображения в представлении. То есть является посредником между слоями.

Сущности (Entities) – слой, содержащий объекты данных, управление которыми – задача только интерактора. Интерактор никогда не передает эти сущности в презентуемый слой, то есть в презентер. Сущности для прередачи за модельный слой должны быть представлены как объекты используемы в презентуемом слое.

Роутер (Router) – слой, ответственный за переходы между модулями VIPER. Роутер может взаимодействовать только с презентером. Презентер вызывает метод роутера и может передать конфигурацию, необходимую для открываемого

модуля, в свою очередь роутер передает эту конфигурацию новому модулю в презентер.

Представление (View) – слой, который отвечает только за пользовательский интерфейс. Представление является пассивным. В данном слое нет обработки данных, представление делегирует эти задачи презентеру. Представление ожидает от презентера контент для отображения и никогда не запрашивает данные у последнего. Презентер в свою очередь не знает о существовании графических элементов, которые находятся в представлении. Презентер знает только о контенте, которым он управляет и времени, когда нужно отобразить или обновить этот контент.

Одним модулем VIPER является один экран приложения.

VIPER по сравнению с паттернами MV(X) имеет несколько отличий в распределении обязанностей. Логика из модели (Model) находится в интеракторе, при этом присутствуют сущности (Entities), которые представляют из себя структуры данных. Презентер в VIPER берет на себя обязанности представления графического интерфейса, вместо контроллера/презентера/представления модели в MVC/MVP/MVVM соответственно. Но в презентере нет возможности изменять данные. В VIPER появляется проблема навигации между модулям, которая решается слоем роутера.

VIPER с точки зрения признаков хорошей архитектуры:

- распределение: обязанности распределены между 5 слоями, что говорит о более широком распределении обязанностей по сравнению с паттернами MV(X);
- тестируемость: тестируемость так же лучше в сравнении с MV(X) благодаря лучшему распределению;
- простота использования: из-за большой распределенности обязанностей увеличивается необходимое количество кода, большое количество интерфейсов/протоколов для классов с незначительными обязанностями.

1.3. Методы взаимодействия клиент-сервер

Взаимодействие клиентского приложения с сервером может осуществляться посредством http-запросов или сокетов.

1.3.1. HTTP-запросы

Протокол http описывает взаимодействие между клиентом и сервером, основанное на сообщениях, которые называются запрос и ответ. Запрос – это сообщение, посылаемое от клиента на сервер. Ответ – сообщение, посылаемое сервером клиенту [18].

Каждое сообщение состоит из 3 частей:

- стартовая строка;
- заголовки;
- тело.

Стартовые строки для запроса и ответа имеют различный формат.

Для запроса стартовая строка должна выглядеть как Метод URI HTTP/Версия, где метод –это метод http-запроса, URI – идентификатор ресурса, версия – версия протокола.

Метод – последовательность символов, за исключением управляющих разделителей. Метод определяет операцию, которую необходимо осуществить с указанным ресурсом [12].

URI (унифицированный идентификатор ресурса) – путь до конкретного ресурса, над которым необходимо осуществить перацию. В случае, когда запросы не относятся к какому-либо конкретному ресурсу, вместо URI указывают “*”. Такой запрос относится к самому веб-сервису [18].

Версия указывает версию стандарта http, например, 1.1.

Стартовая стока для ответа выглядит как HTTP/Версия Код состояния Пояснение.

Версия имеет тоже назначение, что и в запросе.

Код состояния – код, состоящий из трех цифр, первая из которых указывает на класс состояния. В спецификации http версии 1.1 определено 40 кодов

состояния, при этом разрешается расширить протокол и использовать дополнительные коды состояний [18].

Пояснение – текстовое сообщение, поясняющее код ответа, которое предназначено для упрощения прочтения ответа человеком [18].

Заголовки – набор пар имя:значение. Заголовки предназначены для передачи служебной информации [12].

В теле сообщения передаются данные. Данные могут быть в формате xml, html-страницы, JSON.

Методы:

- GET
- POST
- PUT
- PATCH
- DELETE

Метод GET применяется для получения информации. В случае, когда URI ссылается на процесс, который выдаёт данные, то ответом на запрос будут являться данные.

POST используется для отправки данных на сервер, как правило, для создания ресурса. Сервер принимает информацию, которая включена в запрос, как субординарную для ресурса, который указан в статусе в поле URI-запроса.

Метод PUT – применяется для замены уже существующего ресурса. В случае, если URI-запроса ссылается на ресурс, который уже существует, то тело запроса рассматривается как именная версия этого ресурса. Если же URI-запроса ссылается на не существующий ресурс, то такой ресурс создается.

PATCH – так же применяется для изменения существующего ресурса, однако в отличие от PUT, он не заменяет существующий ресурс, а частично изменяет его.

DELETE используется для удаления ресурса.

1.3.2. Формат JSON

JSON (JavaScript Object Notation) – формат обмена данными.

Он является не зависимым от языка реализации. JSON представляет из себя коллекцию пар ключ: значение или упорядоченный список значений. Такие структуры данных присутствуют почти во всех современных языках.

Объект в нотации JSON это неупорядоченный набор пар ключ: значение.

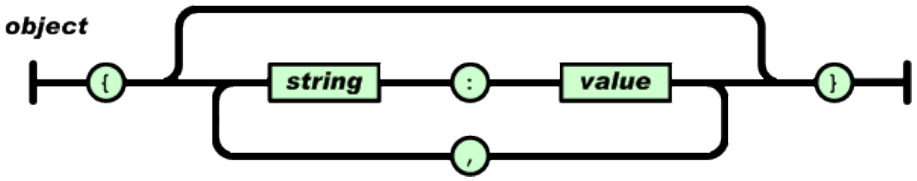

Рисунок 1.3.2.1 – объект в нотации JSON [19]

Массив представляет из себя упорядоченную коллекцию значений.

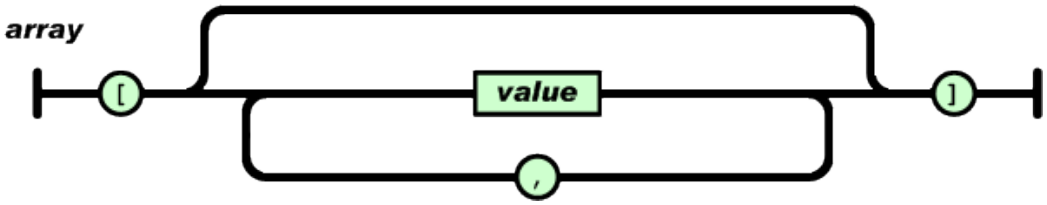

Рисунок 1.3.2.2 – массив в нотации JSON [19]

Значение может быть строкой, числом, true/false, объектом или массивом.

1.3.3. Взаимодействие через сокеты

Сокет – программный интерфейс, предназначенный для обеспечения взаимодействия между процессами. Эти процессы могут протекать как на одном устройстве, так и на двух удаленных устройствах [20].

Сокеты позволяют передавать данные в двух направлениях. Каждый участник данного взаимодействия идентифицируется по двум элементами: Ip-адрес и порт. Первый – идентифицирует устройство, второй – связан с процессом.

Сервер можно рассматривать как процесс, который прослушивает сокет на наличие соединений от клиентов. Каждый процесс может создать сокет, который будет прослушивать канал, привязанный к какому-нибудь порту операционной системы.

В разработке приложений на платформе iOS часто применяется Socket.IO.

Это клиент, находящийся в свободном доступе, который предназначен для реализации взаимодействия клиент-сервер. Представляет из себя протокол,

позволяющий легко использовать Web-сокеты при разработке приложений на платформе iOS.

2. ТРЕБОВАНИЯ К СИСТЕМЕ

Требования к системе были построены на основе поставленных задач и дизайна интерфейса. Полученные требования были представлены в виде вариантов использования.

2.1. Варианты использования

2.1.1. ВИ состояний приложения

Рисунок 2.1.1 Диаграмма ВИ состояний приложения

2.1.1.1. ВИ «Свернуть»

Цель: Изменить состояние приложения на «В бэкграунде».

Начальное состояние: приложение открыто.

Сценарий:

- **a.** пользователь переводит приложение в бэкграунд;
- **b.** система изменяет метод определения координат пользователя на менее чувствительный и продолжает отправлять координаты на сервер при изменении местоположения на 10 метров.

2.1.1.2. ВИ «Закрыть»

Цель: Изменить состояние приложения на «Закрыто».

Начальное состояние: приложение открыто или свернуто.

Сценарий:

- **a.** пользователь удаляет приложение из списка активных приложений;
- **b.** система изменяет метод определения координат на менее чувствительный к изменению местоположения и продолжает отправлять координаты на сервер при изменении местоположения на 100 метров.

2.1.2. ВИ «Вход»/«Регистрация»

Рисунок 2.1.2 – Диаграмма ВИ «Вход»/«Регистрация»

2.1.2.1. ВИ «Продолжить»

Цель: перейти на экран подтверждения номера телефона.

Начальное состояние: приложение открыто, пользователь ввел номер телефона.

Сценарий:

a. пользователь отдает команду «Продолжить»;

b. система показывает экран подтверждения номера телефона.

2.1.2.2. ВИ «Отправить телефонный номер»

Цель: подтвердить введённый номер телефона и отправить его на сервер.

Начальное состояние: пользователь ввел номер телефона и отдал команду «Продолжить».

Сценарий:

a. пользователь отдает команду «Подтвердить»;

b. система отправляет номер телефона на сервер:

- в случае успешного завершения система сохраняет номер телефона и переходит на экран ввода СМС-кода;
- в случае ошибки система показывает сообщение об этой ошибке.

2.1.2.3. ВИ «Отправить СМС-код»

Цель: ввести полученный СМС-код и отправить его на сервер для подтверждения номера телефона.

Начальное состояние: пользователь получил СМС сообщение с кодом для подтверждения номера телефона.

Сценарий:

a. пользователь вводит полученный СМС-код;

b. пользователь отдает команду «Продолжить»;

c. система отправляет СМС-код и номер телефона на сревер для подтверждения номера телефона:

- в случае успешного завершения система открывает экран:
  - «Создать профиль», если пользователь проходит процедуру регистрации;
  - «Карта», если пользователь проходит процедуру авторизации.
- в случае шибки система показывает сообщение об этой ошибке.

2.1.2.4. ВИ «Установить аватар»

Цель: Установить аватар профиля (не является обязательным для завершения процедуры регистрации).

Начальное состояние: телефонный номер подтвержден.

Сценарий:

a. пользователь отдает команду «Установить аватар»;

b. система показывает стандартное для iOS систем контекстное меню для выбора изображения;

c. пользователь выбирает изображение;

d. система отправляет изображение на сервер:

- в случае успешного завершения система отображает установленный аватар;
- в случае ошибки система показывает сообщение об этой ошибке.

2.1.2.5. ВИ «Сохранить»

Цель: завершить процедуру регистрации и создания нового профиля.

Начальное состояние: открыт экран «Создать профиль», пользователь установил аватар и ввел имя (оба пункта не являются обязательными для завершения процедуры регистрации).

Сценарий:

a. пользователь отдает команду «Сохранить»;

b. система отправляет профиль на сервер:

- в случае успеха система открывает главный экран приложения;
- в случае ошибки система показывает сообщение об этой ошибке.

2.1.3. ВИ «Главный экран»

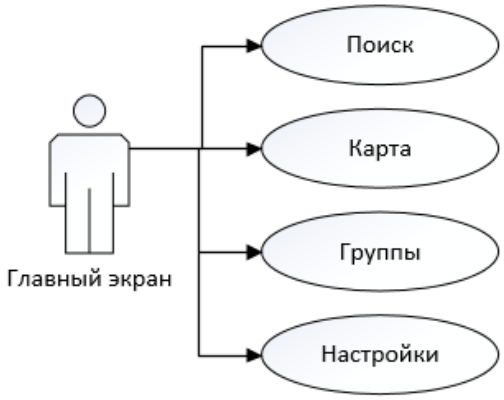

Рисунок 2.1.3 – Диаграмма ВИ «Главный экран»

2.1.3.1. ВИ «Поиск»

Цель: открыть экран «Поиск»

Начальное состояние: открыт экран «Карта», «Группы» или «Настройки».

Сценарий:

a. пользователь отдает команду «Поиск» на панели главного экрана;

b. система открывает экран «Поиск» с результатами последнего поискового запроса.

2.1.3.2. ВИ «Карта»

Цель: открыть экран «Карта».

Начальное состояние: открыт экран «Поиск», «Группы» или «Настройки».

Сценарий:

a. пользователь отдает команду «Карта» на панели главного экрана;

b. система открывает экран «Карта», с картой, отцентрованной на местоположении.

2.1.3.3. ВИ «Группы»

Цель: открыть экран «Группы».

Начальное состояние: открыт экран «Поиск», «Карта» или «Настройки».

Сценарий:

a. пользователь отдает команду «Группы» на панели главного экрана;

b. система открывает экран «Группы».

2.1.3.4. ВИ «Настройки»

Цель: открыть экран «Настройки».

Начальное состояние: открыт экран «Поиск», «Карта» или «Группы».

Сценарий:

a. пользователь отдаёт команду «Настройки» на панели главного экрана;

b. система открывает экран «Настройки».

2.1.4. ВИ «Поиск»

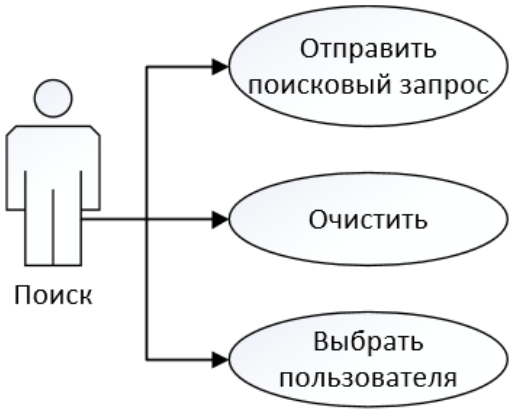

Рисунок 2.1.4 – Диаграмма ВИ «Поиск»

2.1.4.1. ВИ «Отправить поисковый запрос»

Цель: получить результат по введённому поисковому запросу.

Начальное состояние: открыт экран «Поиск», пользователь ввел поисковый запрос.

Сценарий:

a. пользователь отдает команду «Поиск»;

b. система отправляет поисковый запрос на сервер:

- в случае успешного завершения система перезагружает таблицу с результатами поиска;
- в случае ошибки система показывает сообщение об этой ошибке.

2.1.4.2. ВИ «Очистить»

Цель: очистить поле поискового запроса и результаты предыдущего поиска.

Начальное состояние: пользователь ввел поисковый запрос.

Сценарий:

a. Пользователь отдает команду «Очистить»;

b. система очищает поле поискового запроса и таблицу результата поиска.

2.1.4.3. ВИ «Выбрать пользователя»

Цель: открыть экран «Профиль» отображающий данные профиля выбранного пользователя.

Начальное состояние: поиск выполнен успешно, результаты поиска загружены и отображены в таблице результатов поиска.

Сценарий:

a. пользователь выбирает пользователя в таблице результатов поиска;

b. система открывает экран «Профиль» отображающий данные профиля выбранного пользователя.

2.1.5. ВИ «Карта»

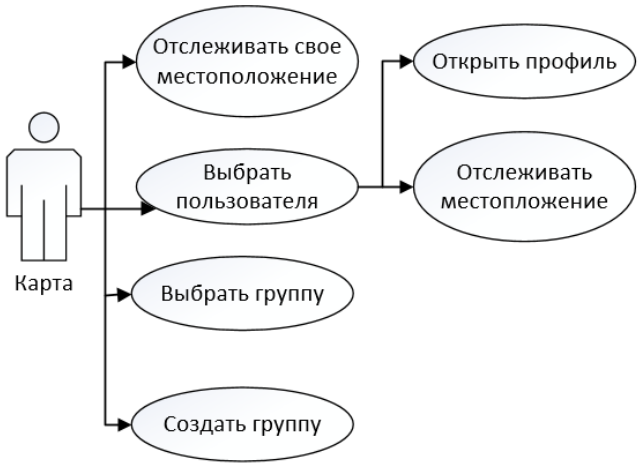

Рисунок 2.1.5 – ВИ «Карта»

2.1.5.1. Use case «Отслеживать свое местоположение»

Цель: отслеживать свое местоположение на карте.

Начальное состояние: открыт экран «Карта».

Сценарий:

a. пользователь отдает команду «Отслеживать вое местоположение»;

b. система начинает изменять положение карты в зависимости от изменения местоположения пользователя.

2.1.5.2. ВИ «Выбрать пользователя»

Цель: открыть всплывающее окно с информацией о выбранном пользователе.

Начальное состояние: открыт экран «Карта», местоположение другого пользователя отображено на карте.

Сценарий:

a. пользователь выбирает другого пользователя путем нажатия на его аннотацию на карте;

b. система открывает всплывающее окно с информацией о профиле выбранного пользователя;

c. система отображает кнопку «Отслеживать местоположение выбранного пользователя».

2.1.5.3. ВИ «Открыть профиль»

Цель: открыть экран «Профиль» с информацией о профиле выбранного пользователя.

Начальное состояние: открыт экран «Карта», выбран пользователь, всплывающее окно с информацией о профиле пользователя отображено.

Сценарий:

a. пользователь отдает команду открыть профиль путем нажатия по всплывающему окну с информацией о профиле;

b. система открывает экран «Профиль» с информацией о профиле выбранного пользователя.

2.1.5.4. ВИ «Отслеживать местоположение»

Цель: отслеживать изменения местоположения выбранного пользователя.

Начальное состояние: открыт экран «Карта», выбран пользователь.

Сценарий:

a. пользователь отдает команду «Отслеживать местоположение выбранного пользователя»;

b. система начинает изменять положение карты в зависимости от изменения местоположения выбранного пользователя.

2.1.5.5. ВИ «Выбрать группу»

Цель: отобразить местоположение на карте членов выбранной группы.

Начальное состояние: открыт экран «Карта».

Сценарий:

a. пользователь выбирает группу на панели группы на экране «Карта»;

b. система удаляет с карты все аннотации членов остальных групп и добавляет аннотации пользователей выбранной группы.

2.1.5.6. ВИ «Создать группу»

Цель: открыть экран «Создать группу».

Начальное состояние: открыт экран «Карта».

Сценарий:

a. пользователь отдает команду «Создать группу»;

b. система открывает экран «Создать группу».

2.1.6. ВИ «Группы»

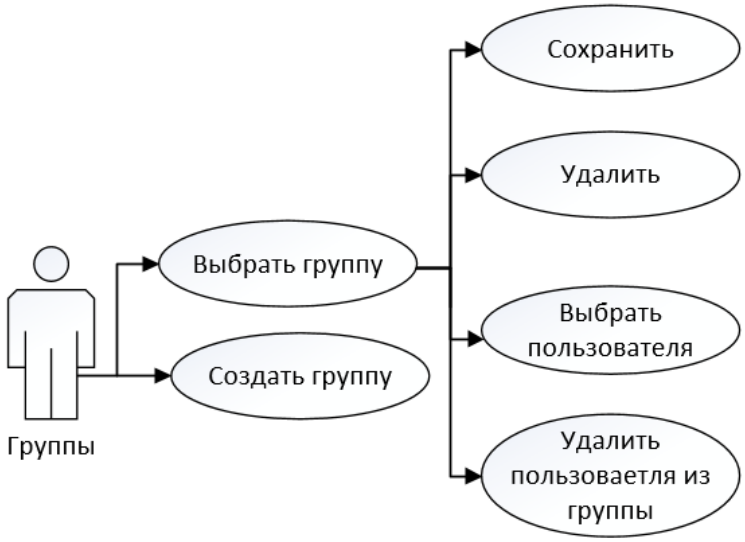

Рисунок 2.1.6 – ВИ «Группы»

2.1.6.1. ВИ «Выбрать группу»

Цель: открыть экран выбранной группы с информацией о группе.

Начальное состояние: открыт экран «Группы».

Сценарий:

a. пользователь отдает команду выбирает группу из коллекции групп на экране «Группы»;

b. система загружает информацию о выбранной группе:

- в случае успешного завершения система открывает экран выбранной группы с информацией о ней
- в случае ошибки система показывает сообщение об этой ошибке.

2.1.6.2. ВИ «Удалить»

Цель: удалить группу.

Начальное состояние: открыт экран выбранной группы, выбранная группа не является одной из стандартных неизменяемых групп («Семья, «Друзья» или «Черный список»).

Сценарий:

a. пользователь отдает команду «Удалить группу»;

b. система отправляет запрос на удаление группы на сервер:

- в случае успешного завершения система закрывает текущий экран и открывает экран «Группы» с обновленной коллекцией групп;
- в случае ошибки система показывает сообщение об этой ошибке.

2.1.6.3. ВИ «Сохранить»

Цель: сохранить изменения в группе.

Начальное состояние: открыт экран выбранной группы.

Сценарий:

a. пользователь удалил члена(ов) группы, изменил имя (изменение имени группы не доступно для стандартных групп) или видимость и отдал команду «Сохранить»;

b. система отправляет изменённую группу на сервер:

- в случае успешного завершения система закрывает текущий экран и открывает экран «Группы» с измененной коллекцией групп;
- в случае ошибки система показывает сообщение об этой ошибке.

2.1.6.4. ВИ «Выбрать пользователя»

Цель: открыть экран «Профиль» с информацией о профиле выбранного пользователя.

Начальное состояние: открыт экран выбранной группы.

Сценарий:

a. пользователь выбирает пользователя из списка членов группы;

b. система открывает экран «Профиль» с информацией о профиле выбранного пользователя.

2.1.6.5. ВИ «Удалить пользователя из группы»

Цель: удалить пользователя из группы.

Начальное состояние: открыт экран выбранной группы, загружены и отображены члены группы.

Сценарий:

a. пользователь отдает команду «Удалить члена группы»;

b. система удаляет соответствующего пользователя из списка членов группы.

2.1.7. ВИ «Создать группу»

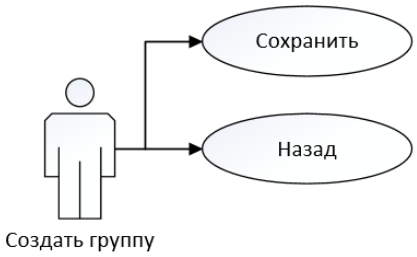

Рисунок 2.1.7 – ВИ «Создать группу»

2.1.7.1. ВИ «Сохранить»

Цель: создать новую группу.

Начальное состояние: открыт экран «Создать группу», пользователь ввел название группы в поле «Название группы», установил видимость группы.

Сценарий:

a. пользователь отдает команду «Сохранить»;

b. система отправляет новую группу на сервер:

- в случае успешного завершения система закрывает текущий экран и открывает предыдущий экран с обновленной коллекцией групп («Карта» - если инициация процесса создания группы начата на этом экране или «Группы» - в другом случае);
- в случае ошибки система показывает сообщение об этой ошибке.

2.1.7.2. ВИ «Назад»

Цель: отменить создание новой группы и вернуться на предыдущий экран.

Начальное состояние: открыт экран «Создать группу».

Сценарий:

a. пользователь отдает команду «Назад»;

b. система закрывает текущий экран и открывает предыдущий.

2.1.8. ВИ «Настройки»

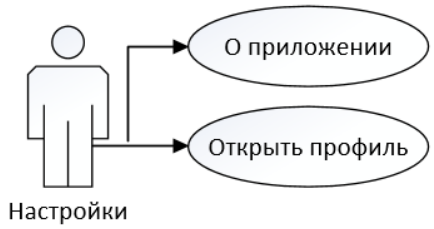

Рисунок 2.1.8 – ВИ «Настройки»

2.1.8.1. ВИ «О приложении»

Цель: открыть экран «О приложении».

Начальное состояние: открыт экран «Настройки».

Сценарий:

a. пользователь выбирает пункт «О приложении»

b. система открывает экран «О приложении».

2.1.8.2. ВИ «Профиль»

Цель: открыть экран профиль с информацией о профиле пользователя.

Начальное состояние: открыт экран «Настройки».

Сценарий:

a. пользователь выбирает пункт «Профиль»;

b. система открывает экран «Профиль» с информацией о профиле пользователя.

2.1.9. ВИ «Профиль»

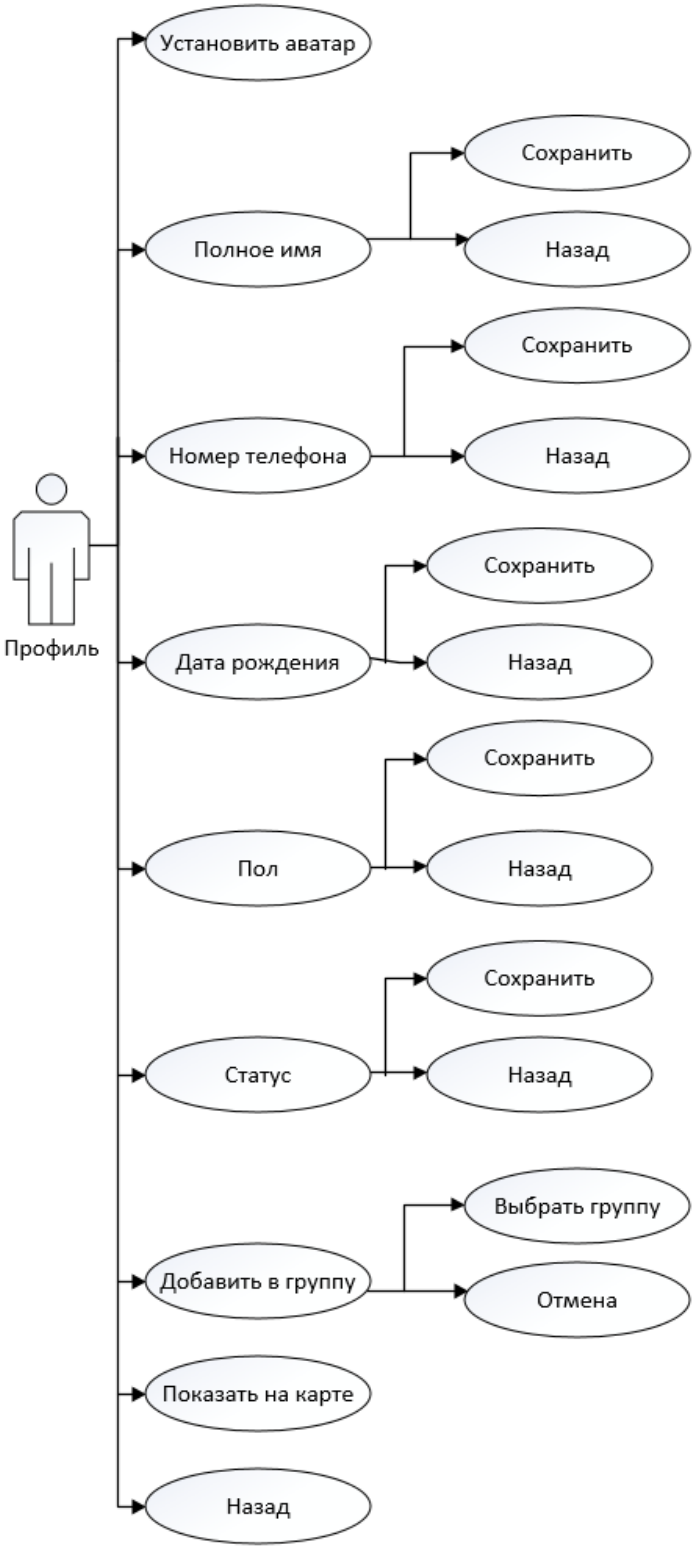

Рисунок 2.1.9 – Диаграмма ВИ «Профиль»

2.1.9.1. ВИ «Установить аватар»

См. п. 2.1.2.4

2.1.9.2. ВИ «Полное имя»

Цель: открыть экран «Полное имя».

Начальное состояние: открыт экран «Профиль».

Сценарий:

a. пользователь выбирает пункт «Имя»;

b. система открывает экран «Полное имя».

2.1.9.3. ВИ «Номер телефон»

Цель: открыть экран «Номер телефона».

Начальное состояние: открыт экран «Профиль».

Сценарий:

a. пользователь выбирает пункт «номер телефона»;

b. система открывает экран «Номер телефона».

2.1.9.4. ВИ «Дата рождения»

Цель: открыть экран «Дата рождения».

Начальное состояние: открыт экран «Профиль».

Сценарий:

a. пользователь выбирает пункт «Дата рождения»;

b. система открывает экран «Дата рождения».

2.1.9.5. ВИ «Пол»

Цель: открыть экран «Пол».

Начальное состояние: открыт экран «Профиль».

Сценарий:

a. пользователь выбирает пункт «Пол»;

b. система открывает экран «Пол».

2.1.9.6. ВИ «Добавить в группу»

Цель: добавить пользователя в группу.

Начальное состояние: открыт экран «Профиль», загружен профиль другого пользователя.

Сценарий:

a. пользователь отдает команду «Добавить в группу»;

b. система открывает экран «Добавить в группу» с загруженной коллекцией групп;

c. пользователь выбирает группу;

d. система отправляет запрос на добавление пользователя в выбранную группу:

- в случае успешного завершения система закрывает текущий экран и открывает экран «Профиль»:
- в случае ошибки система показывает сообщение об этой ошибке.

2.1.9.7. ВИ «Показать на карте»

Цель: показать местоположение пользователя на карте.

Начальное состояние: открыт экран «Профиль», загружен профиль другого пользователя.

Сценарий:

a. пользователь отдает команду «Показать на карте»;

b. система открывает экран «Карта» с картой отцентрованной по местоположению пользователя.

2.1.9.8. ВИ «Сохранить»

Цель: сохранить изменения.

Начальное состояние: открыт экран «Полное имя», «Номер телефона», «Пол», «Дата рождения» или «Статус», изменен параметр профиля и/или его видимость.

Сценарий:

a. пользователь отдает команду «Сохранить»;

b. система отправляет запрос на сохранение изменений на сервер:

- в случае успешного завершения система закрывает текущий экран и открывает экран «Профиль» с обновленной информацией профиля;
- в случае ошибки система показывает сообщение об этой ошибке.

2.1.9.9. ВИ «Назад»

Цель: закрыть текущий экран, вернуться на предыдущий экран, отменить все изменения.

Начальное состояние: открыт экран «Полное имя», «Номер телефона», «Пол», «Дата рождения» или «Статус».

Сценарий:

a. пользователь отдает команду «Назад»;

b. система закрывает текущий экран, открывает экран «Профиль».

3. РЕАЛИЗАЦИЯ

3.1. Архитектура

Архитектура проекта в целом представляет из себя два приложения: мобильное приложение на платформе iOS и серверное приложение.

Серверное приложение обеспечивает обработку данных о местоположении пользователей и сохранение этих данных в БД. Так же сервер отвечает за хранение данных профилей пользователей, которые так же хранятся в БД. Для хранения загруженных пользователями фотографий используется сервис Amazon AWS.

Мобильное приложение посредством взаимодействия через API отправляет данные о местоположении пользователя на сервер, отправляет и получает данные о профиле пользователя, о его группах пользователей, получает данные о профилях других пользователях. Данные об изменении местоположения других пользователей мобильное приложение получает от сервера через web-сокеты.

Архитектура системы в целом представлена на рисунке 3.1.

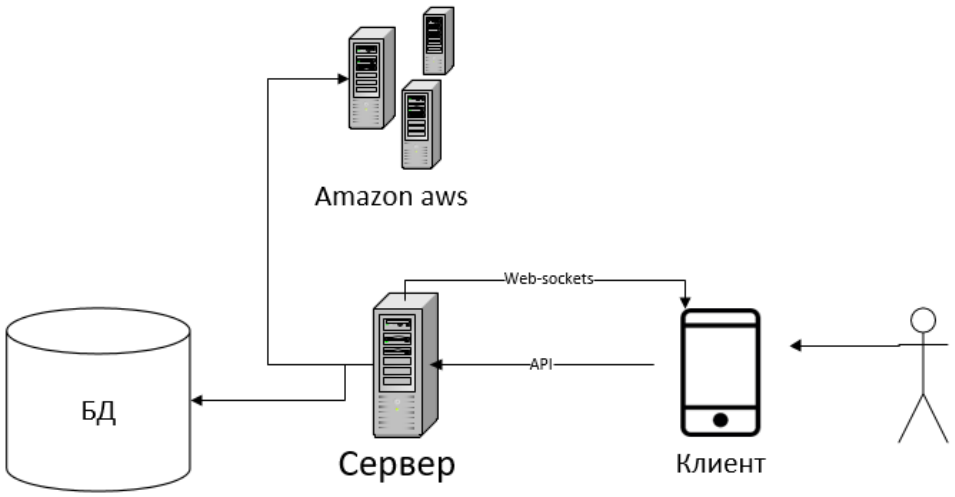

Рисунок 3.1 – архитектура системы

Архитектура мобильного приложения основывается на архитектуре VIPER. Это популярный архитектурный паттерн, схема представлена на рисунке 3.2.

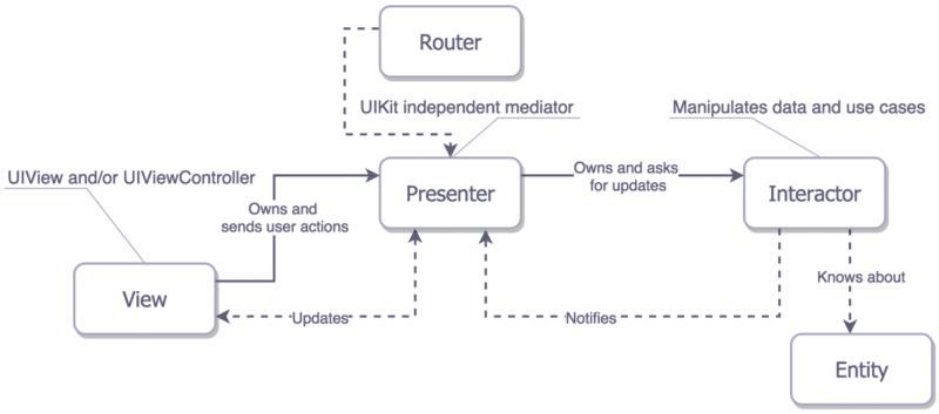

Рисунок 3.2 – схема VIPER

Модулем VIPER может являться один экран или даже целая user story (например, процедура аутентификации может быть на одном или нескольких экранах и вынесена в один модуль).

Частная реализация модуля VIPER, примененная в разработке мобильного приложения, представлена на рисунке 3.3 в виде диаграммы классов.

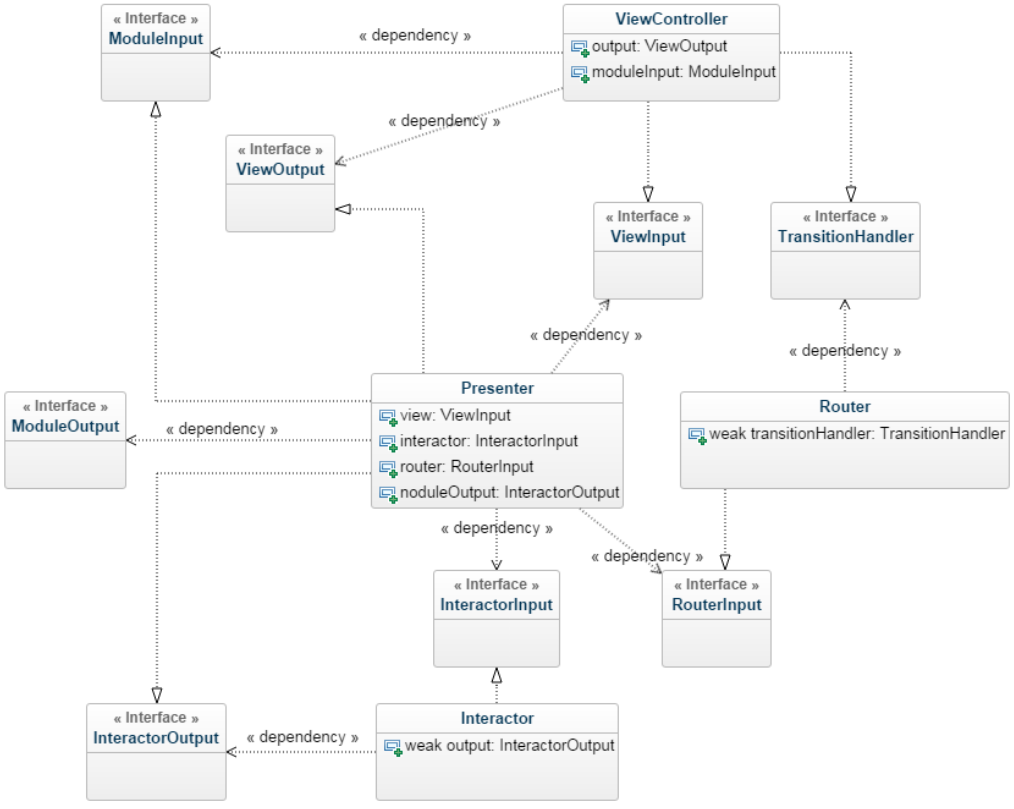

Рисунок 3.3 –Диаграмма классов одного модуля VIPER

Presenter – центральная часть модуля, которая является реализацией протокола.

Общие описания каждого элемента модуля сведены в таблицу 3.1.

Таблица 3.1 – Описание элементов модуля VIPER

View Controller	Обрабатывает все действия пользователя и изменяет интерфейс пользователя
View Input	Интерфейс для доступа к View Controller и Presenter
View Output	Интерфейс для доступа к Presenter из View Controller
Presenter	Отвечает за логику модуля, логику связанную с View
Module Input	Интерфейс для доступа к модулю извне
Module Output	Интерфейс для доступа к другому модулю, который реализует данный интерфейс
Router	Отвечает за переходу между модулями
Router Input	Интерфейс для доступа к Router из Presenter
Transition Handler	Интерфейс для доступа к View Controller из Router
Interactor	Содержит бизнес-логику и имеет доступ к бизнес слою
Interactor Input	Интерфейс для доступа к Interactor из Presenter
Interactor Output	Интерфейс для доступа к Presenter из Interactor

В сравнении с паттернами серии MV(X) VIPER имеет несколько отличий:

- из Модели логика перемещается в Интерактор, Сущности – структуры данных;
- из Контроллера/Представления модели ответственность за отображение пользовательского интерфейса без изменения данных перемещается в Презентер;
- у VIPER лучшее разделение обязанностей между элементами модуля;
- благодаря лучшему распределению обязанностей лучшая тестируемость.

В таблице 3.2 сведены описания элементов модуля для экрана «Профиль».

Таблица 3.2 – Описание элементов модуля «Профиль»

ProfileViewController	Обрабатывает все действия пользователя и оповещает об этом ProfilePresenter, по командам последнего обновляет интерфейс.
ProfileViewInput	ProfileViewController реализует данный интерфейс. Для оповещения ProfileViewController об изменениях в профиле пользователя или для конфигурации элементов экрана ProfliePresenter имеет переменую данного типа. Например, для профиля другого пользователя ProfilePresenter оповещает ProfileViewController о том, что нужно сделать доступными кнопки «Показать на карте» и «Добавить в группу» и сделать недоступной кнопку загрузки аватара.
ProfileViewOutput	ProfilePresenter реализует данный интерфейс. При вызове методов жизненного цикла ProfileViewController, вызываются методы объекта, реализующего данный интерфейс для оповещения ProfilePresenter’а.
ProfilePresenter	Реализует логику модуля и логику обновления интерфейса
ProfileModuleInput	ProfilePresenter реализует данный интерфейс. Перед открытием экрана «Профиль», в модуль может быть передан профиль другого пользователя. В зависимости от наличия профиля другого пользователя в ProfilePresenter конфигурируется интерфейс пользователя.
ProfileRouter	Отвечает за переходу между модулями

ProfileRouterInput	ProfileRouter реализует данный интерфейс. Интерфейс описывает методы взаимодействия с другими модулями.
ProfileInteractor	Содержит бизнес-логику и имеет доступ к бизнес слою. Обращается к объекту сервиса профиля для запроса локальной версии собственного профиля и к объекту сервиса для загрузки, удаления или изменения аватара профиля.
ProfileInteractorInput	ProfileInteractor реализует данный интерфейс. В интерфейсе описаны методы загрузки профиля, загрузки, изменения и удаления аватара профиля.
ProfileInteractorOutput	ProfilePresenter реализует данный интерфейс. Интерфейс описывает методы оповещения ProfilePresenter’а о результатах выполнения методов ProfileInteractor’а.

На рисунке 3.4 приведена диаграмма последовательностей для процесса сохранения изменений имени пользователя.

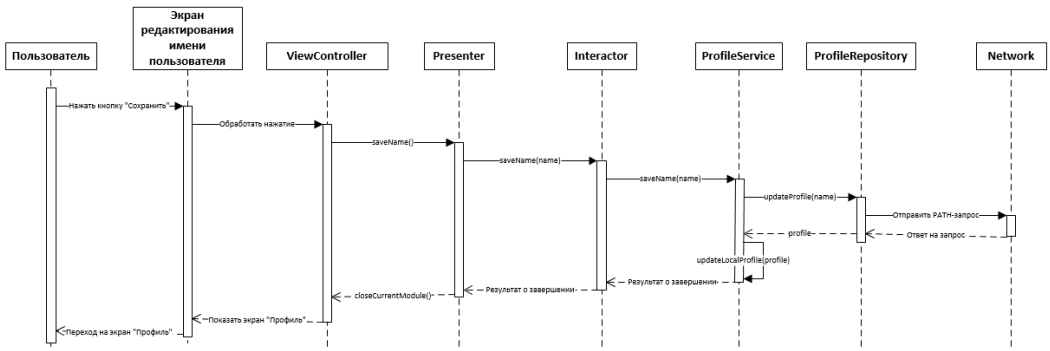

Рисунок 3.4 – диаграмма последовательностей процесса сохранения изменений имени пользователя

3.2. Библиотеки для взаимодействия с сервером

В проекте были использованы следующие библиотеки для работы с сервером: OMGHTTPURL, PromiseKit, Soket.IO и ObjectMapper.

3.2.1. OMGHTTPURL

OMGHTTPURL – это расширение к стандартному протоколу NSURLRequest от Apple, которое инкапсулирует два базовых элемента запроса: URL и политику, используемую при кешировании содержимого URL.

Протокол NSURLRequest был разработан расширяемым для других протоколов для возможности добавления категорий, предоставляющих методы доступа для собственных специфических переменных.

OMGHTTPURL это расширение для протокола NSURLReuest. Он оборачивает конфигурацию запроса в простые методы, которые доступны для разработчиков. Это делает код чище и более читабельным. Нет необходимости писать множество строк кода, конфигурируя запрос, нужно только указать ключевые параметры.

Например, нужно написать следующие строки для отправки POST-запроса на данную URL и с данным JSON:

```
let request = NSMUtableRequest()
request = OMGHTTPURLQ.post(urlString, json: json)
```

OMGHTTPURL является частью PromiseKit.

3.2.2. PromiseKit

PromiseKit предоставляет Промисы (Promises), которые очень эффективны при работе с асинхронными задачами. PromiseKit обернул в Промисы почти все Apple API.

Все задачи связанные с сетью являются асинхронными, что делает PromiseKit идеальным решением для работы с подобными задачами.

Для отправки и обработки запросов в проекте использовались Сервисы и Репозитории.

Для отправки запроса вызывается метод сервиса, в котором вызывается метод в репозитории, в котором формируется данные для отправки запроса и вызывается метод класса, ответственного за отправку запросов, с входными параметрами сформированными в этом методе.

Так как согласно архитектурному паттерну VIPER Интерактор отвечает за бизнес-логику, то вызов метода сервиса происходит в Интеракторе.

```
    import PromiseKit
    class ExampleInteractor: ExampleInteractorInput {
          weak var output: ExampleInteractorOutput?
    let exampleService = ExampleServiceType
          init(exampleService: ExampleServiceType) {
               self.exampleService = exampleService
          }
          func sendSomeString(exampleString: String) {
               firstly {
                    exampleService.sendString(exampleString:
exampleString)
               }.then { response in
                    self.handle(response)
               }.catch { error in
                    self.output?.showError(error.message())
               }.always {
                    self.alwaysExampleMethod()
               }
          }
    }
```

У метода «sendSomeString» параметром является объект типа String. В протоколе «ExampleServiceType» есть метод «sendString». «exampleService» - это объект типа «ExampleService» (класс, реализующий протокол «ExampleServiceType»). Метод «sendString» возвращает объект типа «URLDataPromise» или просто Промис. В ходе последовательного вызова всех методов формируется цепочка из возвращаемых этими методами Промисов, которая начинается с момента отправки запроса и заканчивается в Интеракторе. Если на каком-то из этапов этой цепи появляется ошибка, то она автоматически оказывается в конце цепи, то есть в Интеракторе.

Блок «then» вызывается в случае, когда не возникло никаких ошибков цепочке Промисов. Если метод сервиса «exampleService» возвращает какой-либо объект, он должен быть обработан в Интеракторе, и только потом Презентер должен быть оповещен об этом.

Блок «catch» вызывается в случае возникновения ошибки в цепочке Промисов. Презентер реализует протокол «ExampleInteractorOutput», поэтому в случае вызова «catch» Интерактор информирует Презентер об возниконовении

этой ошибки. Как пример, Интерактор может вызвать метод Презентера, передав текст ошибки как параметр, после чего Презентер вызовет метод Представления, так же передав текст ошибки как параметр, для того чтобы представление инициировал показ информационного сообщения с текстом ошибки.

Блок «always» вызывается не зависимо от факта возникновения ошибки в цепочке Промисов. Как правило вызов этого блока используется для остановки анимации индикатора загрузки на экране приложения. В таком случае внутри блока может быть вызван метод Презентера, внутри которого будет вызван метод Представления, который останавливает анимацию индикатора загрузки.

3.2.3. Soket.IO

Socket.IO-ios-swift это soket-клинет, используемы в приложениях, разраббатываемых на платформе iOS и написанный языке программирования Swift. Для взаимодействия с сервером он использует Web-сокеты.

Взаимодействие посредством web-сокетов опирается на клиент-серверную логику, где соединение между клиентом и сервером существует всегда.

В данном проекте сокеты были использованы для обновления местоположения пользователей на карте в режиме реального времени.

Использование данного клиента весьма простое.

Для начала необходимо создать объект клинета.

```
let socket = SocketIOClient(socketURL: "youtServerURLString:8900")
```

После этого необходимо объявить обработчики событий.

```
socket.on("connect") { data, ack in
      self.socket.emit("listen user", with [token])
}
socket.on("message") { data, ack in
      //notify delegate
}
```

Метод «on» вызывается, когда socket-клиент получает сообщение от сервера с соответствующим тегом. Например, при инициализации соединения между клиентом и сервером, клиент получит сообщение с тегом «connect». В теле данного метода вызывается метод отправки сообщения («emit») на сервер. В методе «emit» указываются тег «listen user», по которому сервер определит

нужный обработчик, и объект, который будет отправлен на сервер. В данном случае устанавливается соединение между сервером и клиетом, подтверждением, авторизацией является отправка токена на сервер.

Когда клиент получает сообщение от сервера, то внутри соответствующего метода «on» должен быть вызван метод делагата объекта socket-клиента.

Для установления соединения методы «on» и вызов метода socket-клинета «connect()» должны быть обернуты в единый метод.

```
func startListeningForEvents() {
      socket.on("connect") {…}
      socket.on("message") {…}
      socket.connect()
}
```

Для разрыва соединения между клиентом и сревером нужно вызвать метод «disconnect()»

```
func stopListening() {
      socket.disconnect()
}
```

3.2.4. ObjectMapper

ObjectMapper это фреймворк, написанный на языке программирования Swift, для трансформации модели в JSON или наоборот.

Когда с сервера приходят данные, они должны быть трансформированы в модели для дальнейшего использования в приложении.

Структуры имеют преимущества над классами в качестве моделей, потому что они относительно занимают меньше пространства в памяти и они копируются. Копирование объектов более безопасно чем множественные ссылки на тот же инстанс, что происходит, если в качестве объектов модели используется классы. Это особенно важно в процессах обхода переменных и/или в многопоточной среде. Если всегда отправлять копию объекта, то не нужно беспокоиться о том, что этот объект изменений в каком-то другом месте и возникнет конфликт.

При использовании структур вероятность появления утечек памяти в разы меньше.

Для трансформации JSON в модель, используемую в приложении структура должна реализовывать протокол «Mappable» или «ImmutableMappable», являющиеся частью фреймворка ObjectMapper.

Протокол «Mappable» используется в общих случаях. Протокол «ImmutableMappable» может быть использван в случаях, когда есть 100% гарантия, что определенные поля передаваемого объекта будут не пустыми.

```
struct User{
      var id: Int
      var firstName: String?
      var lastName: String?
      var photo: Photo?
}
extension User: ImmutableMappable {
      init(map: Map) throws {
            id = try map.value("userId")
      }
      mutating func mapping(map: Map) {
            id        <- map["userId"]
            firstName <- map["first_name"]
            lastName  <- map["last_name"]
            photo          ,_ map["photo"]
      }
}
```

Это пример структуры, описывающей объект, который будет получен в результате трансформации из JSON. Этот процесс называется «маппингом».

Прежде всего объявляются поля структуры «User», такие как id, имя, фамилия и фото. Эти поля могут иметь любой тип данных, даже тип другой структуры.

Язык программирования Swift предоставляет возможность использовать опциональный тип данных, называемый optional type (Например, String?). Переменные такого типа могут иметь значение, а могут быть nil. Объект типа optional должен быть разыменовании перед использованием.

```
var optionalValue: String? = "some string"
var someString = "optional string is going to be after this text" + optionalValue!
```

Поля, которые могут не иметь значения объявляются с опциональными типами данных. Поле «id» не должно быть опциональным, так как его значение

уникально и идентифицирует пользователя в системе и есть 100% гарантия, что с сервера придет объект, у которого это поле не будет nil.

Следующим шагом является расширение структуры «User» имплементацией протокола «ImmutableMappable». В конструкторе «init» должна быть инициализирована переменная «id», так как ее тип не является optional type. Данный конструктор может возвращать ошибку, которая прервет цепочку Промисов.

Метод «mapping» связывает значения из JSON-объекта со значениями структуры «User» по ключам (userId, first_name, last_name, photo).

Трансформирование данных при использовании PromiseKit:

```
class ExampleRepoSitory: ExampleRepositoryType {
      let mapping: ExampleMappingType
      init(mapping: ExampleMappingType) {
            self.mapping = mapping
      }
      func mapExample() {
            //send some GET request which returns data
            firstly {…
            }.then { data in
                  self.mapping.map(data: data)
            }
      }
}
```

4. РЕЗУЛЬТАТ

На рисунках 4.1 – 4.2 приведены скриншоты процесса регистрации/входа.

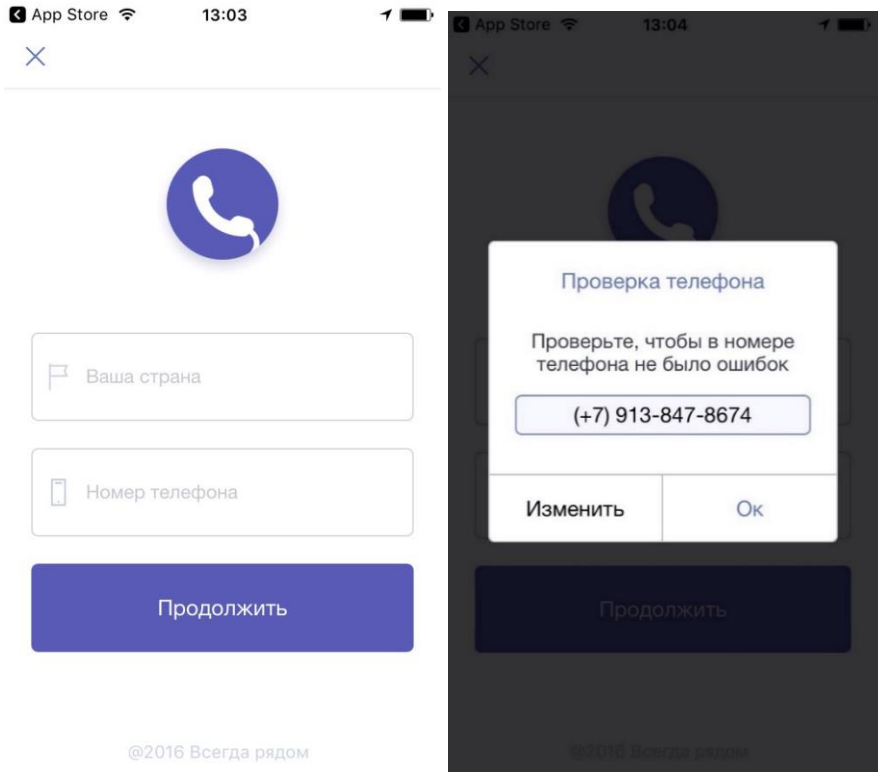

Рисунок 4.1 – экран ввода номера телефона (слева), экран подтверждения номера телефона (справа)

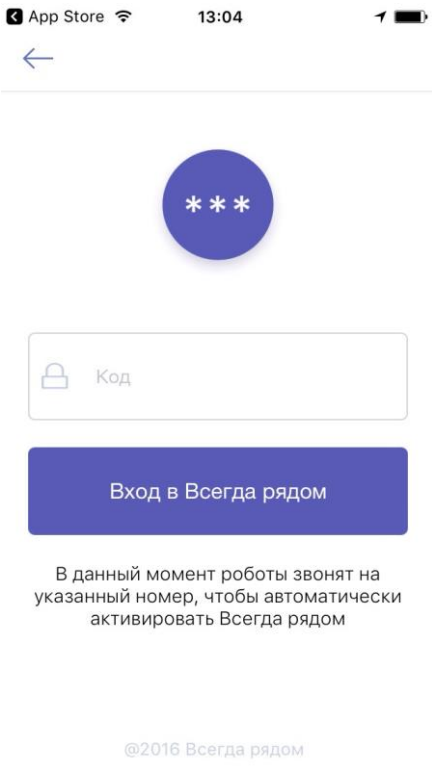

Рисунок 4.2 – экран ввода СМС-кода

На рисунках 4.3 – 4.4 представлен главный экран приложения с картой.

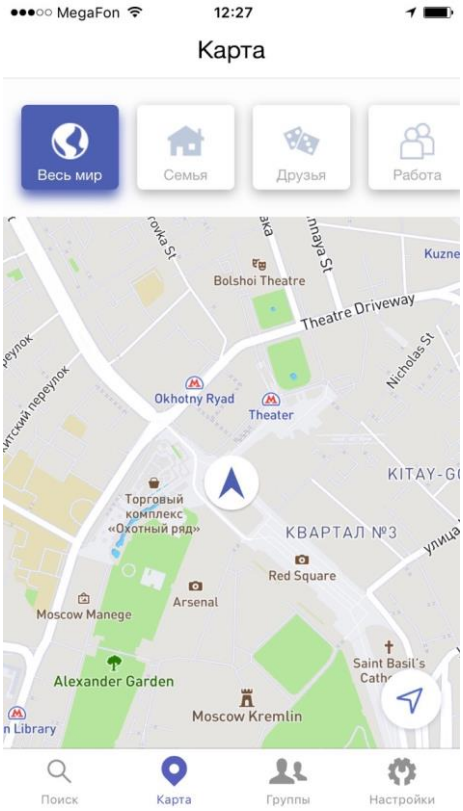

Рисунок 4.3 –главный экран с картой

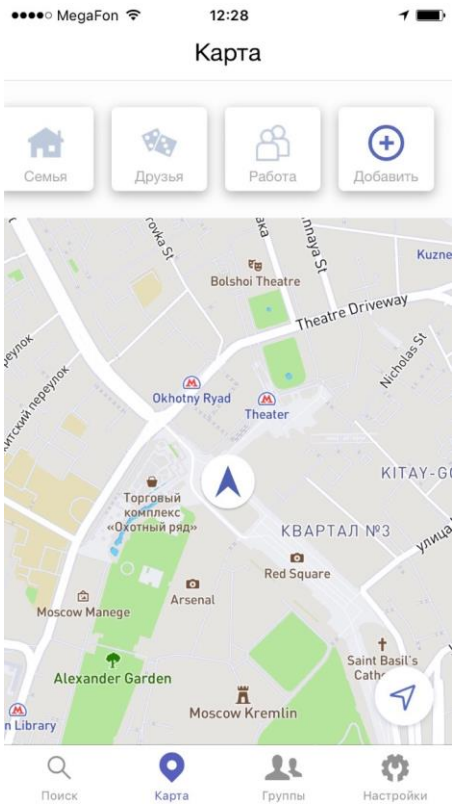

Рисунок 4.4 –главный экран с картой

На рисунках 4.5 – 4.6 приведены скриншоты экрана «Поиск».

Рисунок 4.5 – экран «Поиск»

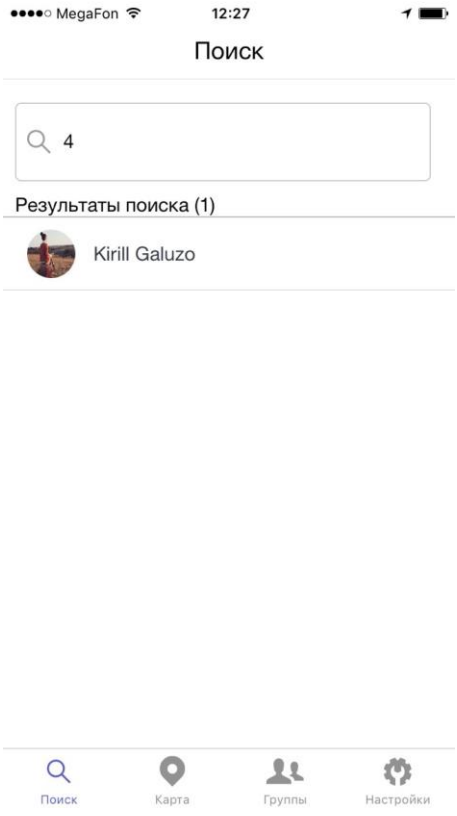

Рисунок 4.6 – экран «Поиск»

На рисунках 4.7 – 4.8 приведены скриншоты экрана «Группы» и экрана для отдельной группы.

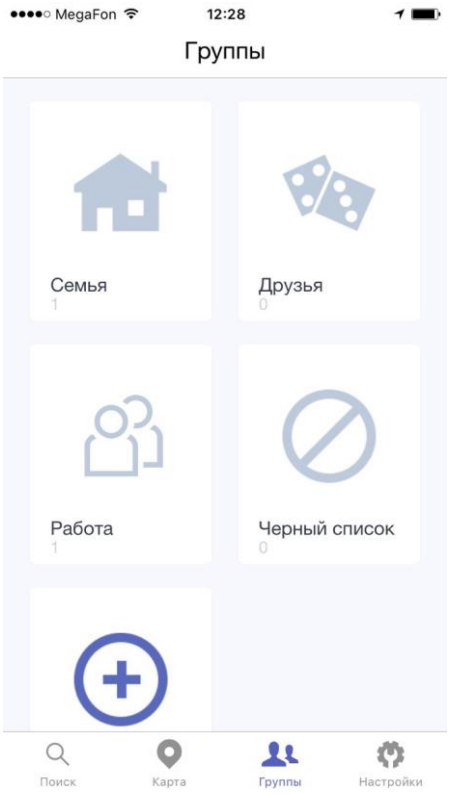

Рисунок 4.7 – экран «Группы»

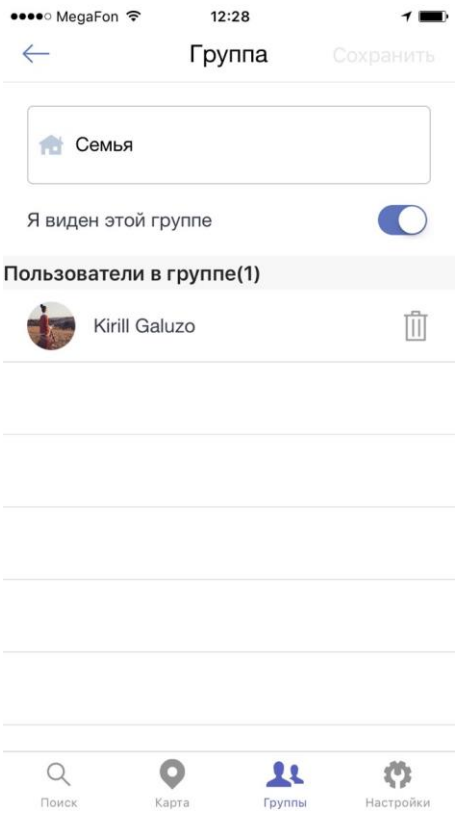

Рисунок 4.8 – экран группы «Семья»

На рисунке 4.9 представлен экран «Настройки».

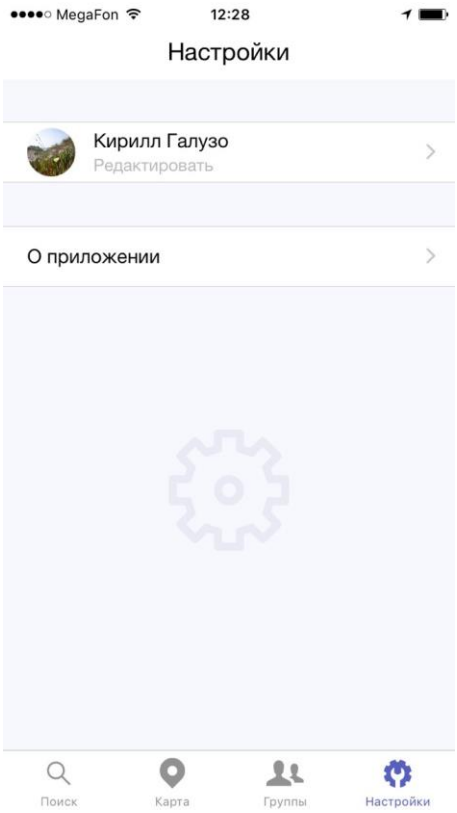

Рисунок 4.9 – экран «Настройки»

Скриншоты на рисунках 4.10 – 4.15 демонстрируют экран «Профиль» и экраны для просмотра/изменения настроек профиля.

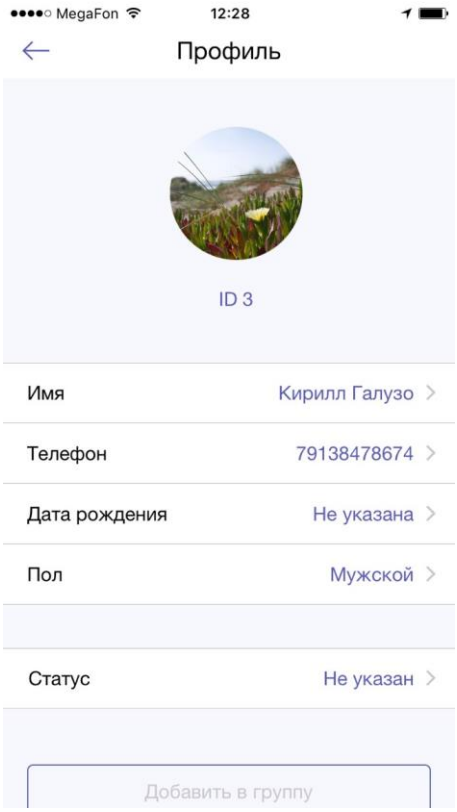

Рисунок 4.10 – экран «Профиль»

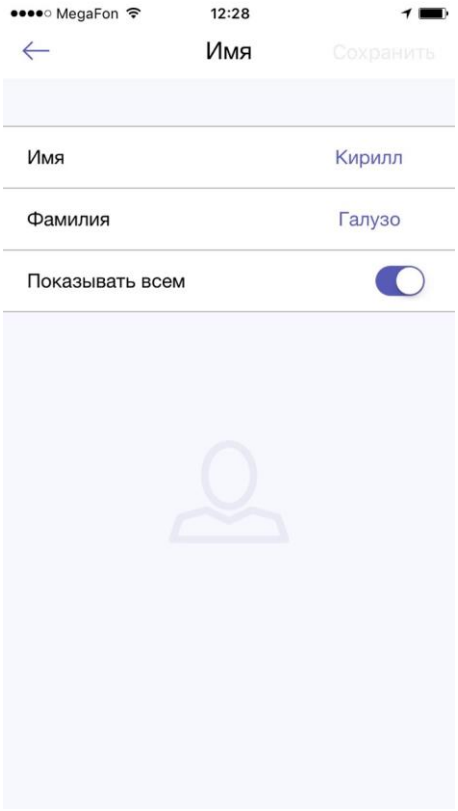

Рисунок 4.11 – экран редактирования имени пользователя

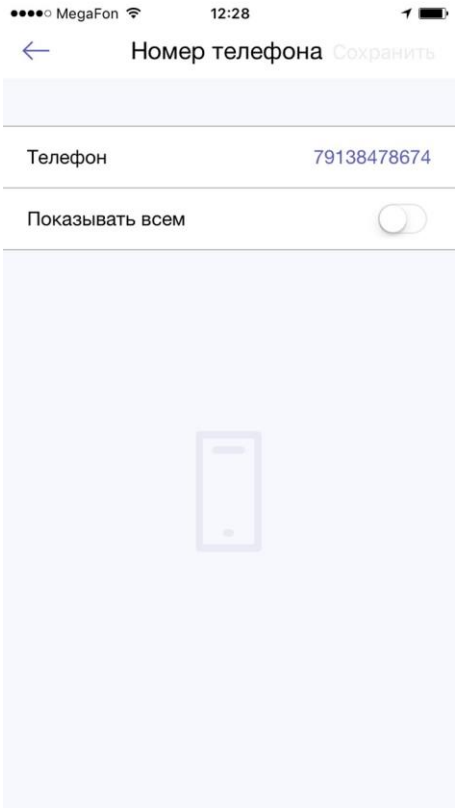

Рисунок 4.12 – экран редактирования настроек номера телефона

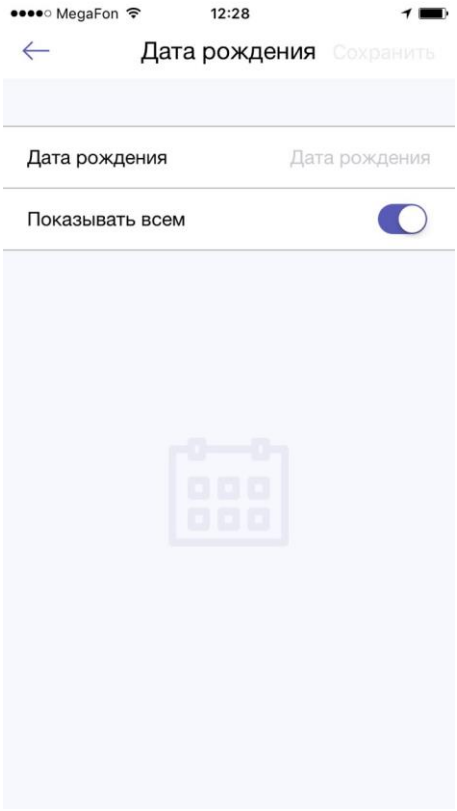

Рисунок 4.13 – экран редактирования даты рождения

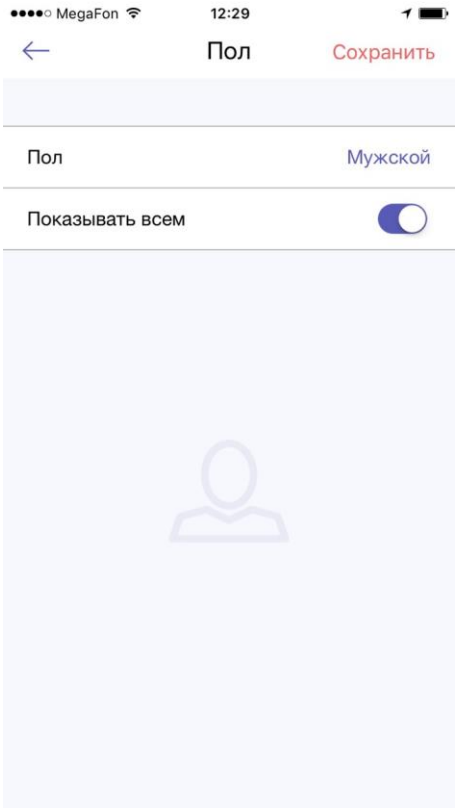

Рисунок 4.14 – экран редактирования пола

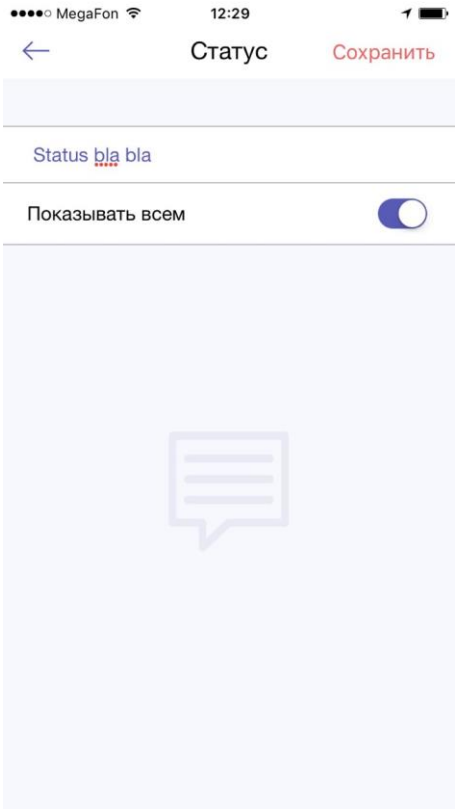

Рисунок 4.15 – экран редактирования статуса

5. РАЗДЕЛ «ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Целью данного раздела является определение оценки коммерческого потенциала, перспективности и альтернатив разработки программного продукта с позиции ресурсоэффективности и ресурсосбережения, а также планирование и формирование бюджета научных исследований, определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования.

Разработанный программный продукт предназначен для использвания лицами старше 4х лет для обмена данными о своем местоположении в режиме реального времени.

5.1.Организация и планирование работ

При организации работ в рамках проектной работы необходимо планировать занятость каждого участника проекта в работе. На данном этапе определяется полный перечень работ, распределение времени работ между всеми участниками.

5.2.Продолжительность этапов работ

Расчет продолжительности этапов работ может осуществляется опытно-статистическим методом. Для расчета ожидаемого значения продолжительности работ $t_{ож}$ применяются две оценки: $t_{\min}$ и $t_{\max}$ (метод двух оценок).

$$t_{ож} = \frac{3t_{\min} + 2t_{\max}}{5},$$

где $t_{\min}$ – минимальная трудоемкость работ, чел/дн;

$t_{\max}$ – максимальная трудоемкость работ, чел/дн.

Для выполнения разработки требуется два исполнителя (исполнитель 1 и исполнитель 2).

Для построения линейного графика рассчитывается длительность этапов в рабочих днях, а затем осуществляется её перевод в календарные дни. Расчёт

продолжительности выполнения каждого этапа в рабочих днях ($T_{РД}$) выполняется по формуле:

$$T_{РД} = \frac{t_{ож}}{K_{ВН}} \cdot K_{Д},$$

где $t_{ож}$ – продолжительность работы, дн.;

$K_{ВН}$ – коэффициент выполнения работ ($K_{ВН} = 1$);

$K_{Д}$ – коэффициент, учитывающий дополнительное время на компенсацию непредвиденных задержек и согласование работ ($K_{Д} = 1,2$).

Расчёт продолжительности этапа в календарных днях осуществляется по формуле:

$$T_{КД} = T_{РД} \cdot T_{К},$$

где $T_{КД}$ – продолжительность выполнения этапа в календарных днях;

$T_{РД}$ – продолжительность выполнения этапа в рабочих днях;

$T_{К}$ – коэффициент календарности.

Коэффициент календарности рассчитывается по формуле:

$$T_{К} = \frac{T_{КАЛ}}{T_{КАЛ} - T_{ВД} - T_{ПД}},$$

где $T_{КАЛ}$ – календарные дни, $T_{КАЛ} = 365$;

$T_{ВД}$ – выходные дни, $T_{ВД} = 52$;

$T_{ПД}$ – праздничные дни, $T_{ПД} = 10$.

Подставив значения в формулу, получим следующий результат:

$$T_{К} = \frac{365}{365 - 52 - 10} = 1,205.$$

В таблице 5.1 приведена длительность этапов работ и число исполнителей, занятых на каждом этапе.

Таблица 5.1 – Временные показатели проведения научного исследования

Этап	**Исполнители**	**Продолжительность работ, дни**			**Длительность работ, чел/дн.**			
					$T_{РД}$		$T_{КД}$	
		t_{min}	t_{max}	$t_{ож}$	**И1**	**И2**	**И1**	**И2**
1. Формирование ТЗ	Исполнитель1, Исполнитель 2	2	5	3,6	4,32	4,32	5,21	5,21
2. Анализ предметной области, сбор и анализ исходных данных	Исполнитель 1, Исполнитель 2	3	6	4,2	5,04	5,04	6,07	6,07
3. Проектирование архитектуры приложения	Исполнитель 1	2	5	3,6	4,32	0	5,21	0
4. Изучение методов определения местоположения устройствами, работающими на платформе iOS	Исполнитель 1	1	4	2	2,4	0	2,89	0
5. Изучение методов взаимодействия клиент-сервер	Исполнитель 1	1	4	2	2,4	0	2,89	0
6. Разработка программного продукта	Исполнитель 1, Исполнитель 2	50	70	58	69,6	54,8	83,87	66
7. Тестирование разработанного продукта	Исполнитель 1, Исполнитель 2	5	10	7	2,52	5,88	3,04	7,09
8. Приемка программного продукта	Исполнитель 1, Исполнитель 2	4	8	5,6	6,72	3,56	8,1	4,29
Итого:		**68**	**112**	**86**	**97,32**	**73,6**	**117,28**	**88,66**

5.3.Разработка графика проведения научного исследования

Для наглядного отображения графика и распределения работ между участниками проекта использована диаграмма Ганта. Диаграмма Ганта представляет собой ленточный график, на котором работы по теме представляются протяженными во времени отрезками, характеризующеся датами начала и окончания выполнения того или иного этапа работ.

Календарный план-график проведения работ представлен в таблице 5.2.

Таблица 5.2 – Календарный план-график проведения работ

№ раб	Вид работ	$T_{кд}$ И1	$T_{кд}$ И2	Продолжительность выполнения работ														
				Февраль			Март			Апрель			Май			Июнь		
				1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	Формирование ТЗ	4,32	4,32															
2	Анализ предметной области, сбор и анализ исходных данных	5,04	5,04															
3	Проектирование архитектуры приложения	4,32	0															
4	Изучение методов определения местоположения устройствами, работающими на платформе iOS	2,89	0															
5	Изучение методов взаимодействия клиент-сервер	2,89	0															
6	Разработка программного продукта	83,87	66															
7	Тестирование разработанного продукта	3,04	7,09															
8	Приемка программного продукта	8,1	4,29															

Примечание к таблице 5.2: - - Исполнитель 1

- Исполнитель 2

5.4.Бюджет разработки

В состав бюджета выполнения работ по разработке мобильного приложения включает вся себя стоимость всех расходов, необходимых для их выполнения. При формировании бюджета используется группировка затрат по следующим статьям:

- материалы и покупные изделия;
- заработная плата;
- отчисления в страховые фонды;
- накладные расходы.

В соответствии с текущими требованиями налогово-бухгалтерской отчетности амортизационные отчисления со вновь приобретенных объектов взымаются от стоимости единицы 100000 рублей. Если стоимость оборудования ниже этой величины, ее относят на материальные затраты.

5.4.1. Расчет материальных затрат

Так как все работы выполнялись преимущественно на компьютерном оборудовании от корпорации Apple, то потребовались затраты на приобретение двух компьютеров Mac-Mini, а так же на канцелярские принадлежности и флэш карту. Все материальные затраты отображены в таблице 5.3.

Таблица 5.3 – Материальные затраты

Наименование	Ед. изм.	Количество	Цена за ед., руб.	Затраты на материалы, (**Зм**), руб.
Бумага формата А4	Уп.	1	150	150
Чернила	Шт.	1	250	250
Флэш-карта	Шт.	1	500	500
Комптьютер Mac-mini	Шт.	2	55000	110000
Итого				**110900**

Транспортно-заготовительные расходы (ТРЗ) составляют 5% от отпускной цены материалов. Расходы на материалы с учётом ТРЗ:

$$C_{MAT} = 110900 \cdot 1{,}05 = 116445 \text{ руб.}$$

5.4.2. Расчет заработной платы

Данная статья расходов включает заработную плату двух исполнителей. Расчет основной заработной платы выполняется на основе трудоёмкости выполнения каждого этапа и величины месячного оклада исполнителя.

Работы выполнялись на основании договора гражданского правового характера (хоз. договора).

Затраты времени по каждому исполнителю в рабочих днях взяты из таблицы 1. Для перехода от тарифной суммы заработка исполнителя, связанной с участием в проекте, к соответствующему полному заработку необходимо будет тарифную сумму заработка исполнителя, связанной с участием в проекте умножить на интегральный коэффициент. Интегральный коэффициент находится по формуле:

$$К_и = К_{пр} \cdot К_{доп.ЗП} \cdot К_р,$$

где Кпр – коэффициент премий, Кпр = 1,1;

Кдоп.ЗП – коэффициент дополнительной зарплаты при пятидневной рабочей неделе Кдоп.ЗП = 1,113;

Кр – коэффициент районной надбавки, Кр = 1,3.

Результаты вычислений представлены в таблице 5.4.

Таблица 5.4 - Основная заработная плата исполнителей системы

Исполнитель	**Вознаграждение, руб./мес**	$ЗП_{дн-т}$, **руб./раб.день**	**Затраты времени, раб.дни**	**Коэффициент**	**Фонд з/платы, руб.**
И1	25200	1200	98	1,59	186984
И2	25200	1200	74	1,59	141192
Итого:					328176

5.4.3. Расчет отчислений в страховые фонды

Взнос в социальные фонды установлен в размере 30,2% от заработной платы. Размер взноса рассчитываются по формуле:

$$С_{соц} = С_{ЗП} \cdot 0{,}302,$$

где СЗП – размер заработной платы.

Подставив необходимые значения в формулу и получим:

$$С_{соц} = 328176 \cdot 0{,}302 = 99109{,}15 \text{ руб.}$$

5.4.4. Расчет накладных расходов

Прочие расходы следует принять равными 20% от суммы всех предыдущих расходов. Они находятся по формуле:

$$С_{накл} = (С_{ЗП} + С_{соц}) \cdot 0,$$

Где $С_{ЗП}$ – основная заработная плата, руб.;

$С_{соц}$ – отчисления в страховые фонды, руб.;

Подставив полученные выше результаты, получим:

$$С_{накл} = (328176 + 99109{,}15) * 0{,}2 = 85457{,}03 \text{ руб.}$$

5.4.5. Расчет общей стоимости разработки

Проведя расчет по всем статьям сметы затрат на разработку, можно определить общую себестоимость проекта.

Таблица 5.5 – Смета затрат на разработку проекта

Статья затрат	Условное обозначение	Сумма, руб.
Материалы и покупные изделия	$С_{мат}$	116445
Основная заработная плата	$С_{ЗП}$	328176
Отчисления в социальные фонды	$С_{соц}$	99109,15
Прочие расходы	$С_{накл}$	85457,03
Итого:		**629187,18**

Таким образом, затраты на разработку составили С =629187,18руб.

5.5.Оценка экономической эффективности

Выполнение разработки оценивается уровнями достижения экономического, научного, научно-технического и социального эффектов.

Для итоговой оценки результатов проекта в зависимости от поставленных целей в качестве критерия эффективности принимается один из видов эффекта, а остальные используются в качестве дополнительных характеристик.

На данном этапе внедрение нет возможности оценить экономический эффект в количественных показателях. Так как данная разработка является мобильным приложением находящемся в свободном доступе, для дальнейшего расширения функционала и монетизации расширенного функционала. Следовательно, в дальнейшем необходимо рассчитывать данный показатель исходя из заявленных параметров и условий. Поэтому в качестве критерия эффективности проекта оценим научно-технический уровень НИР.

5.5.1. Оценка научно-технического уровня НИР

Научно-технический уровень характеризует влияние проекта на уровень и динамику обеспечения научно-технического прогресса в данной области. Для оценки научной ценности, технической значимости и эффективности, планируемых и выполняемых НИР, используется метод балльных оценок. Каждому фактору по принятой шкале присваивается определенное количество

баллов. Обобщенная оценка проводится по сумме баллов по всем показателям. На её основе делается вывод о целесообразности НИР.

Интегральный показатель научно технического уровня НИР определяется по формуле:

$$I_{\text{НТУ}} = \sum_{i=1}^{3} R_i \cdot n_i,$$

где $I_{\text{НТУ}}$ – интегральный индекс научно-технического уровня;

R_i – весовой коэффициент i-го признака научно-технического эффекта;

n_i – количественная оценка i-го признака научно-технического эффекта, в баллах.

Весовые коэффициенты признаков НТУ приведены в таблице 5.6.

Таблица 5.6 – Весовые коэффициенты признаков НТУ

Признаки научно-технического эффекта НИР	**Характеристика признака НИР**	$\boldsymbol{R_i}$
Уровень новизны	Систематизируются и обобщаются сведения, определяются пути дальнейших исследований	0,40
Теоретический уровень	Разработка способа (алгоритма)	0,10
Возможность реализации	Время реализации в течение первых лет	0,50

Баллы для оценок уровня новизны, теоретического уровня и возможности реализации приведены в таблицах 7 – 9.

Таблица 5.7 – Баллы для оценки уровня новизны

Уровень новизны	Характеристика уровня новизны	Баллы
Принципиально новая	Новое направление в науке и технике, новые факты и закономерности, новая теория, вещество, способ	8–10
Новая	По-новому объясняются те же факты, закономерности, новые понятия дополняют ранее полученные результаты	5–7
Относительно новая	Систематизируются, обобщаются имеющиеся сведения, новые связи между известными	2–4
Не обладает новизной	Результат, который ранее был известен	0

Таблица 8 – Баллы значимости теоретических уровней

Теоретический уровень полученных результатов	Баллы
Установка закона, разработка новой теории	10
Глубокая разработка проблемы, многоспектральный анализ взаимодействия между факторами с наличием объяснений	8
Разработка способа (алгоритм, программа)	6
Элементарный анализ связей между фактами (наличие гипотезы, объяснения версии, практических рекомендаций)	2
Описание отдельных элементарных факторов, изложение наблюдений, опыта, результатов измерений	0,5

Таблица 9 – Возможность реализации результатов по времени

Время реализации	Баллы
В течение первых лет	10
От 5 до 10 лет	4
Свыше 10 лет	2

В таблице 10 указано соответствие качественных уровней НИР значениям показателя.

Таблица 10– Оценка научно-технического уровня НИР

Фактор НТУ	**Значимость**	**Уровень фактора**	**Выбранный балл**	**Обоснование выбранного балла**
Уровень новизны	0,4	Аналог существующих решений	2	Отличный от аналогов интерфейс, меньшее потребление заряда аккумулятора, повышенная точность определения координат
Теоретический уровень	0,1	Разработка способа	6	Способ обмена данными о местоположении между пользователями
Возможность реализации	0,5	В течение первых лет	10	Быстрая разработка с помощью различных инструментальных средств

Интегральный показатель научно-технического уровня составляет:

$$I_{НТУ} = 0{,}4 \cdot 2 + 0{,}1 \cdot 6 + 0{,}5 \cdot 10 = 6{,}4.$$

Таблица 11 – Оценка уровня научно-технического эффекта

Уровень НТЭ	**Показатель НТЭ**
Низкий	1–4
Средний	4–7
Высокий	8–10

Таким образом, согласно таблице 11, разработка, на тему «Разработка мобильного веб-сервиса для обмена данными о местоположении пользователей» имеет средний уровень научно-технического эффекта.

6. РАЗДЕЛ «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Разработанный в рамках магистерской диссертации проект является мобильным веб-сервисом, который позволяет пользователям обмениваться данными о своем местоположении в режиме реального времени.

Разработка системы велась с использованием компьютерной техники. Использование средств вычислительной техники накладывает ряд вредных факторов на человек, что в последствии снижает производительность его труда и может привести к существенным проблемам здоровья сотрудника.

В данном разделе проведен анализ вредных и опасных факторов производственной среды как для разработчиков, так и для пользователей.

6.1. Производственная безопасность на стадии разработки мобильной веб-системы

Научно-исследовательская деятельность выполнялась в помещении ООО «С Медиа Линк» по адресу проспект Кирова 51а строение 5 офис 513. Помещение оснащено видео-дисплейными терминалами (ВДТ), персональными электронно-вычислительными машинами (ПЭВМ), компьютерными столами, стульями, столом для коллективной работы, огнетушителями, кондиционером, противопожарной сигнализацией и датчиками дыма.

Для обеспечения производственной безопасности необходимо проанализировать воздействия на человека вредных и опасных производственных факторов, которые могут возникать при разработке проекта.

Производственный фактор считается вредным, если воздействие этого фактора на человека может привести к его заболеванию. Производственный фактор считается опасным, если его воздействие может привести к травме [1].

Все производственные факторы классифицируются по группам элементов: физические, химические, биологические и психофизические. Для данной работы целесообразно рассмотреть физические и психофизические вредные и опасные факторы производства, характерные для рабочей зоны

программиста, разработчика приложения, пользователя. Выявленные факторы представлены в таблице 6.1.

Таблица 6.1 – Вредные и опасные производственные факторы при выполнении работ за ПЭВМ

Источник фактора, наименование видов работ	Факторы (по ГОСТ 12.0.003-74)		Нормативные документы
	Вредные	Опасные	
1) Работа за ПК	1) Повышенный уровень шума на рабочем месте; 2) Недостаточная освещенность рабочей зоны; 3) Умственное перенапряжение; 4) Монотонный режим работы.	1) Опасность поражения электрическим током; 2) Опасность возникновения пожара.	1) СН 2.2.4/2.1.8.562-96; 2) СанПиН 2.2.4.548-96; 3) СанПиН 2.2.2/2.4.1340-03; 4) СП 52.13330.2011; 5) ГОСТ Р 12.1.019-2009 ССБТ; 6) СНиП 21-01-97.

6.1.1 Вредные производственные факторы

6.1.1.1. Недостаточная освещенность рабочей зоны

Недостаточная освещенность рабочей зоны является вредным производственным фактором, возникающим при работе с ПЭВМ, уровни которого регламентируются СП 52.13330.2011.

Причиной недостаточной освещенности являются недостаточность естественного освещения, недостаточность искусственного освещения, пониженная контрастность.

Работа с компьютером подразумевает постоянный зрительный контакт с дисплеем ПЭВМ и занимает от 80 % рабочего времени. Недостаточность освещения снижает производительность труда, увеличивает утомляемость и количество допускаемых ошибок, а также может привести к появлению профессиональных болезней зрения.

Разряд зрительных работ программиста и оператора ПЭВМ относится к разряду III и подразряду Г (работы высокой точности). В таблице 6.2 представлены нормативные показатели искусственного освещения при работах заданной точности.

Таблица 6.2 – Требования к освещению помещений промышленных предприятий для операторов ПЭВМ [4]

Характеристика зрительной работы	**Наименьший или эквивалентный размер объекта различения, мм**	**Разряд зрительной работы**	**Подразряд зрительной работы**	**Контраст объекта с фоном**	**Характеристика фона**	**Искусственное освещение**		
						Освещённость, лк		
						При системе комбинированного освещения		**При системе общего освещения**
						всего	В том числе от общего	
Высокой точности	0,264	III	г	Средний, большой	Светлый, средний	400	200	200

Для создания и поддержания благоприятных условий освещения для операторов ПЭВМ, их рабочие места должны соответствовать санитарно-эпидемиологическим правилам СанПиН 2.2.2/2.4.1340-03. Рабочее помещение должно иметь естественное и искусственное освещение, соответствующее показателям, представленным в таблице 6.6.. Для рассеивания естественного освещения следует использовать жалюзи на окнах рабочих помещений. В качестве источников искусственного освещения должны быть использованы люминесцентные лампы, лампы накаливания – для местного освещения [5].

6.1.1.2. Умственное перенапряжение

Умственное перенапряжение вызывается большим объемом информации, которую надо анализировать, и чтобы избежать умственного перенапряжения необходимо устраивать небольшие перерывы в течение рабочего дня продолжительностью не более 5 минут.

При умственной работе, по сравнению с физической работой потребление кислорода мозгом увеличивается в 15-20 раз. Если для умственной работы требуется значительное нервно-эмоциональное напряжение, то возможны значительные изменения кровяного давления, пульса. Длительная работа этого характера может привести к заболеванию, в частности сердечно-сосудистым и некоторым другим заболеваниям [4].

6.1.1.3. Монотонный режим работы

При работе с ПЭВМ основным фактором, влияющим на нервную систему программиста или пользователя, является огромное количество информации, которое он должен воспринимать. Это является сложной задачей, которая очень сильно влияет на сознание и психофизическое состояние из-за монотонности работы. Поэтому меры, позволяющие снизить воздействие этого вредного производственного фактора, которые регулируются СанПиН 2.2.2/2.4.1340-03, являются важными в работе оператора ПЭВМ. Они позволяют увеличить производительность труда и предотвратить появление профессиональных болезней.

Организация работы с ПЭВМ осуществляется в зависимости от вида и категории трудовой деятельности. Виды трудовой деятельности разделяются на 3 группы: группа А – работа по считыванию информации с экрана с предварительным запросом; группа Б – работа по вводу информации; группа В – творческая работа в режиме диалога с ПЭВМ. Работа программиста-разработчика рассматриваемой в данной работе относится к группам А и Б, в то время, как деятельность пользователя приложения относится к группе В. Категории трудовой деятельности, различаются по степени тяжести

выполняемых работ. Для снижения воздействия рассматриваемого вредного фактора предусмотрены регламентированные перерывы для каждой группы работ – таблица 6.3.

Таблица 6.3 – Суммарное время регламентированных перерывов в зависимости от продолжительности работы, вида категории трудовой деятельности с ПЭВМ [5]

Категория работы с ПЭВМ	**Уровень нагрузки за рабочую смену при видах работ с ПЭВМ**			**Суммарное время регламентированных перерывов, мин.**	
	группа А, количество знаков	**группа Б, количество знаков**	**группа В, ч**	**при 8-часовой смене**	**при 12-часовой смене**
I	до 20 000	до 15 000	до 2	50	80
II	до 40 000	до 30 000	до 4	70	110
III	до 60 000	до 40 000	до 6	90	140

Для предупреждения преждевременной утомляемости пользователей ПЭВМ рекомендуется организовывать рабочую смену путем чередования работ с использованием ПЭВМ и без него. В случаях, когда характер работы требует постоянного взаимодействия с компьютером (работа программиста-разработчика) с напряжением внимания и сосредоточенности, при исключении возможности периодического переключения на другие виды трудовой деятельности, не связанные с ПЭВМ, рекомендуется организация перерывов на 10–15 мин. через каждые 45–60 мин. работы. При высоком уровне напряженности работы рекомендуется психологическая разгрузка в специально оборудованных помещениях [5].

6.1.2. Опасные производственные факторы

6.1.2.1. Опасность поражения электрическим током

Поражение электрическим током является опасным производственным фактором и, поскольку программист имеет дело с электрооборудованием, то вопросам электробезопасности на его рабочем месте должно уделяться особое внимание. Нормы электробезопасности на рабочем месте регламентируются

СанПиН 2.2.2/2.4.1340-03, вопросы требований к защите от поражения электрическим током освещены в ГОСТ Р 12.1.019-2009 ССБТ.

Электробезопасность – система организационных и технических мероприятий и средств, обеспечивающих защиту людей от вредного и опасного воздействия электрического тока, электрической дуги, электромагнитного поля и статического электричества.

Опасность поражения электрическим током усугубляется тем, что человек не в состоянии без специальных приборов обнаружить напряжение дистанционно.

Помещение, где расположено рабочее место оператора ПЭВМ, относится к помещениям без повышенной опасности ввиду отсутствия следующих факторов: сырость, токопроводящая пыль, токопроводящие полы, высокая температура, возможность одновременного прикосновения человека к имеющим соединение с землей металлоконструкциям зданий, технологическим аппаратам, механизмам и металлическим корпусам электрооборудования.

Основным организационным мероприятием по обеспечению безопасности является инструктаж и обучение безопасным методам труда, а также проверка знаний правил безопасности и инструкций в соответствии с занимаемой должностью применительно к выполняемой работе.

К мероприятиям по предотвращению возможности поражения электрическим током относятся:

- С целью защиты от поражения электрическим током, возникающим между корпусом приборов и инструментом при пробое сетевого напряжения на корпус, корпуса приборов и инструментов должны быть заземлены;
- При включенном сетевом напряжении работы на задней панели корпуса приборов должны быть запрещены;
- Все работы по устранению неисправностей должен производить квалифицированный персонал;

- Необходимо постоянно следить за исправностью электропроводки [5, 6].

6.1.2.2. Опасность возникновения пожара

Возникновение пожара является опасным производственным фактором, т.к. пожар на предприятии наносит большой материальный ущерб, а также часто сопровождается травмами и несчастными случаями. Регулирование пожаробезопасности производится СНиП 21-01-97.

В помещениях с ПЭВМ повышен риск возникновения пожара из-за присутствия множества факторов: наличие большого количества электронных схем, устройств электропитания, устройств кондиционирования воздуха; возможные неисправности электрооборудования, освещения, или неправильная их эксплуатация может послужить причиной пожара.

Возможные виды источников воспламенения:

- Искра при разряде статического электричества;
- Искры от электрооборудования;
- Искры от удара и трения;
- Открытое пламя [7].

Для профилактики организации действий при пожаре должен проводиться следующий комплекс организационных мер: должны обеспечиваться регулярные проверки пожарной сигнализации, первичных средств пожаротушения; должен проводиться инструктаж и тренировки по действиям в случае пожара; не должны загромождаться или блокироваться пожарные выходы; должны выполняться правила техники безопасности и технической эксплуатации электроустановок; во всех служебных помещениях должны быть установлены «Планы эвакуации людей при пожаре и других ЧС», регламентирующие действия персонала при возникновении пожара.

Для предотвращения пожара помещение с ПЭВМ должно быть оборудовано первичными средствами пожаротушения: углекислотными огнетушителями типа ОУ-2 или ОУ-5; пожарной сигнализацией, а также, в

некоторых случаях, автоматической установкой объемного газового пожаротушения [7].

6.2. Экологическая безопасность

6.2.1. Влияние объекта исследования на окружающую среду

В данном разделе рассматривается воздействие на окружающую среду деятельности по разработке проекта, а также самого продукта в результате его реализации на производстве.

В ходе выполнения ВКР и дальнейшем использовании алгоритмов отсутствуют выбросы каких-либо вредных веществ в атмосферу и гидросферу, следовательно, загрязнение воздуха и воды не происходит.

Люминесцентные лампы, применяющиеся для искусственного освещения рабочих мест, также требуют особой утилизации, т.к. в них присутствует от 10 до 70 мг ртути, которая относится к чрезвычайно-опасным химическим веществам и может стать причиной отравления живых существ, а также загрязнения атмосферы, гидросферы и литосферы. Сроки службы таких ламп составляют около 5-ти лет, после чего их необходимо сдавать на переработку в специальных пунктах приема.

Во время разработки и написания ВКР образовывался мусор, такой как: канцелярские принадлежности, бумажные отходы, неисправные комплектующие персонального компьютера, люминесцентные лампы.

6.2.2. Мероприятия по защите окружающей среды

Для уменьшения вредного влияния на литосферу необходимо производить сортировку отходов и обращаться в службы по утилизации для дальнейшей переработки или захоронения. [14]

В основном, организации, занимающиеся приёмом и утилизацией ртуть содержащих отходов, принимают люминесцентные лампы в массовых количествах. Лампа состоит из электронного блока — выгодный компонент для реставрации и утилизации; колба и цоколь также ценное сырье. По стране

утилизацией «ртутных» ламп занимаются более 50 фирм, но единственное их условие — деньги, которые вы должны заплатить за вывоз.

Такие лампы нельзя выкидывать в мусоропровод или уличные контейнеры, а нужно отнести в свой районный ДЕЗ (Дирекция единичного заказчика) или РЭУ (Ремонтно-эксплуатационное управление), где есть специальные контейнеры. Там они принимаются бесплатно, основанием должна служить утилизация в соответствии с Управлением Федеральной службы по надзору в сфере защиты прав потребителей и благополучия человека по Томской области. Пункты приёма отработавших свой срок люминесцентных ламп по городам можно найти в интернете. [15]

Переработка макулатуры представляет собой многоэтапный процесс, цель которого заключается в восстановлении бумажного волокна и, зачастую, других компонентов бумаги (таких как минеральные наполнители) и использование их в качестве сырья для производства новой бумаги.

Организации, занимающиеся покупкой сломанных компьютеров на запчасти, готовы платить за запчасти деньги, которые они сэкономят на покупке новых деталей, необходимых для ремонта. Такие организации принимают даже битую и залитую чем-то технику. Компьютерная техника (или ее компоненты) может также заинтересовать тех, кто скупает старые платы и радиодетали для получения из них после переработки драгоценных и редких металлов. Многие сетевые гипермаркеты электронной техники периодически устраивают программу утилизации. Условия такие: за старую бытовую технику вам предложат неплохую скидку на последующую покупку в этом магазине. Также можно самостоятельно отвезти сломанный компьютер в пункт приема металлолома не составит труда. Такие точки приема есть в каждом городе

6.3. Безопасность в чрезвычайных ситуациях

6.3.1. Основные чрезвычайные ситуации в офисном помещении

Чрезвычайные ситуации бывают техногенного, природного, биологического, социального или экологического характера.

При работе в кабинете могут возникнуть следующие классификации чрезвычайных ситуаций:

- Преднамеренные/непреднамеренные;
- Техногенные: взрывы, пожары, обрушение помещений, аварии на системах жизнеобеспечения/природные – связанные с проявлением стихийных сил природы.
- Экологические – это аномальные изменения состояния природной среды, такие как загрязнения биосферы, разрушение озонового слоя, кислотные дожди/ антропогенные – являются следствием ошибочных действий людей.
- Биологические – различные эпидемии, эпизоотии, эпифитотии;
- Социальные – это обстановка на определенной территории, сложившаяся в результате опасного социального явления, которое повлекло в результате человеческие жертвы, ущерб здоровью, имуществу или окружающей среды;
- Комбинированные.

6.3.2. Типичные чрезвычайные ситуации

6.3.2.1. Пожар (возгорание)

Наиболее вероятная чрезвычайная ситуация, которая может возникнуть при работе с ПЭВМ – пожар, так как в современных ЭВМ очень высокая плотность размещения элементов электронных схем. В непосредственной близости друг от друга располагаются соединительные провода и кабели, при протекании по ним электрического тока выделяется значительное количество теплоты, при этом возможно оплавление изоляции и возникновение возгорания.

Биологические, так как программист работает в кабинете и контактирует с большим количеством людей, в том числе с другими

сотрудниками, то велик риск заражения одного сотрудника от другого (чем больше народу, тем выше риск). В связи с большим скоплением народа в одном помещении появляется необходимость в непрерывном проветривании, что приводит к образованию сквозняков, что так же может сказаться на здоровье.

Возникновение других видов ЧС – маловероятно [12].

6.3.3. Действия в результате возникновения черезвычайной ситуации и мер по ликвидации последствий

При работе компьютерной техники выделяется много тепла, что может привести к пожароопасной ситуации. Источниками зажигания так же могут служить приборы, применяемые для технического обслуживания, устройства электропитания, кондиционеры воздуха. Серьёзную опасность представляют различные электроизоляционные материалы, используемые для защиты от механических воздействий отдельных радиодеталей.

В связи с этим, участки, на которых используется компьютерная техника, по пожарной опасности относятся к категории пожароопасных «В».

Меры, соблюдение которых поможет исключить с большой вероятностью возможность возникновения пожара:

- Для понижения воспламеняемости и способности распространять пламя кабели покрывают огнезащитным покрытием;
- При ремонтно-профилактических работах строго соблюдаются правила пожарной безопасности;
- Помещения, в которых должны располагаться ПЭВМ проектируют I или II степени огнестойкости;
- Каждое из помещений, где производится эксплуатация устройств ПЭВМ, должно быть оборудовано первичными средствами пожаротушения и обеспечено инструкциями по их применению. В качестве средств пожаротушения разрешается использование углекислотного огнетушителя типа ОУ-2, ОУ-5(описание ниже), а также

порошковый тип. Применение пенных огнетушителей не допускается, так как жидкость пропускает ток;

- Устройства ПЭВМ необходимо устанавливать вдали отопительных и нагревательных приборов (расстояние не менее 1 м и в местах, где не затруднена их вентиляция и нет прямых солнечных лучей);
- Разрабатываются организационные меры по обучению персонала навыкам ликвидации пожара имеющимися в наличии средствами тушения пожара до прибытия пожарного подразделения [8].

При пожаре люди должны покинуть помещение в течение минимального времени.

В помещениях с компьютерной техникой, недопустимо применение воды и пены ввиду опасности повреждения или полного выхода из строя дорогостоящего электронного оборудования.

Для тушения пожаров необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем. Воду разрешено применять только во вспомогательных помещениях [7].

6.4. Правовые и организационные вопросы обеспечения безопасности

6.4.1. Описание правовых норм для работ, связанных с работой на ПЭВМ

Регулирование отношений между работником и работодателем, касающихся оплаты труда, трудового распорядка, особенности регулирования труда женщин, детей, людей с ограниченными способностями и проч., осуществляется законодательством РФ, а именно трудовым кодексом РФ.

Нормальная продолжительность рабочего времени не может превышать 40 часов в неделю.

Порядок исчисления нормы рабочего времени на определенные календарные периоды (месяц, квартал, год) в зависимости от установленной

продолжительности рабочего времени в неделю определяется федеральным органом исполнительной власти, осуществляющим функции по выработке государственной политики и нормативно-правовому регулированию в сфере труда.

Продолжительность ежедневной работы (смены) не может превышать:

- Для работников в возрасте от 15 до 16 лет – 5 часов, в возрасте от 16 до 18 лет – 7 часов;
- Для учащихся общеобразовательных учреждений, образовательных учреждений начального и среднего профессионального образования, совмещающих в течение учебного года учебу с работой, в возрасте от 14 до 16 лет – 2,5 часа, в возрасте от 16 до 18 лет – 4 часов;
- Для инвалидов – в соответствии с медицинским заключением, выданным в порядке, установленном федеральными законами и иными нормативными правовыми актами российской федерации.

Для работников, занятых на работах с вредными и (или) опасными условиями труда, где установлена сокращенная продолжительность рабочего времени, максимально допустимая продолжительность ежедневной работы (смены) не может превышать:

- При 36-часовой рабочей неделе - 8 часов;
- При 30-часовой рабочей неделе и менее - 6 часов.

Продолжительность работы (смены) в ночное время сокращается на один час без последующей отработки. К работе в ночное время не допускаются: беременные женщины; работники, не достигшие возраста 18 лет, за исключением лиц, участвующих в создании и (или) исполнении художественных произведений, и других категорий работников в соответствии с настоящим Кодексом и иными федеральными законами.

В течение рабочего дня (смены) работнику должен быть предоставлен перерыв для отдыха и питания. Время предоставления перерыва и его конкретная продолжительность устанавливаются правилами внутреннего

трудового распорядка или по соглашению между работником и работодателем.

Всем работникам предоставляются выходные дни (еженедельный непрерывный отдых).

Организация-работодатель выплачивает заработную плату работникам. Возможно удержание заработной платы только в случаях, установленных ТК РФ ст. 137. В случае задержки заработной платы более чем на 15 дней, работник имеет право приостановить работу, письменно уведомив работодателя.

Законодательством РФ запрещена дискриминация по любым признакам и принудительный труд [9].

Если пользователь постоянно загружен работой с ЭВМ, приемлемой является поза сидя. В положении сидя основная нагрузка падает на мышцы, поддерживающие позвоночный столб и голову. В связи с этим при длительном сидении время от времени необходимо сменять фиксированные рабочие позы.

Исходя из общих принципов организации рабочего места, в нормативно-методических документах сформулированы требования к конструкции рабочего места.

Основными элементами рабочего места программиста являются: рабочий стол, рабочий стул (кресло), дисплей, клавиатура, мышь; вспомогательными - пюпитр, подставка для ног [10].

Взаимное расположение элементов рабочего места должно обеспечивать возможность осуществления всех необходимых движений и перемещений для эксплуатации и технического обслуживания оборудования [11].

Рабочие места с ЭВМ должны располагаться па расстоянии не менее 1,5 м от стены с оконными проемами, от других стен – на расстоянии 1 м, между собой – на расстоянии не менее 1,5 м. При размещении рабочих мест

необходимо исключить возможность прямой засветки экрана источником естественного освещения.

При размещении ЭВМ на рабочем месте должно обеспечиваться пространство для пользователя величиной не менее 850 м. Для стоп должно быть предусмотрено пространство по глубине и высоте не менее 150 мм, по ширине – не менее 530 мм. Располагать ЭВМ на рабочем месте необходимо так, чтобы поверхность экрана находилась на расстоянии 400 – 700 мм от глаз пользователя. Конструкция рабочего места и взаимное расположение всех его элементов (сиденье, органы управления, средства отображения информации и т.д.) должны соответствовать антропометрическим, физиологическим и психологическим требованиям, а также характеру работы [12].

Рабочее кресло обеспечивает поддержание рабочей позы в положении сидя, и чем длительнее это положение в течение рабочего дня, тем жестче должны быть требования к созданию удобных и правильных рабочих сидений.

Высота поверхности сиденья должна регулироваться в пределах 400 – 550 мм. Ширина и глубина его поверхности должна быть не менее 400 мм. Поверхность сиденья должна быть плоской, передний край – закругленным. Сиденье и спинка кресла должны быть полумягкими, с нескользящим, не электризующимся и воздухопроницаемым покрытием, материал которого обеспечивает возможность легкой очистки от загрязнения.

Опорная поверхность спинки стула должна иметь высоту 280 – 320 мм, ширину – не менее 380 мм и радиус кривизны горизонтальной плоскости – 400 мм. Расстояние сцинки от переднего края сиденья должно регулироваться в пределах 260 – 400 мм.

Рабочее место должно быть оборудовано устойчивой и просто регулируемой подставкой для ног, располагающейся, по возможности, по всей ширине отводимого участка для ног. Подставка должна иметь ширину не менее 300 мм, глубину не менее 400 мм, регулировку по высоте до 150 мм и по углу наклона опорной поверхности подставки до 20. Поверхность

подставки должна быть рифленой, по переднему краю иметь бортик высотой 10 мм.

При организации рабочего пространства необходимо учитывать индивидуальные антропометрические параметры пользователя с соответствующими допусками на возможные изменения рабочих поз и потребность в перемещениях.

Рациональной рабочей позой может считаться такое расположение тела, при котором ступни работника расположены на плоскости пола или на подставке для ног, бедра сориентированы в горизонтальной плоскости, верхние части рук – вертикальный угол локтевого сустава колеблется в пределах 70 – 90, запястья согнуты под углом не более чем 20, наклон головы – в пределах 15 – 20, а также исключены частые ее повороты [10].

6.4.2. Влияние реализации системы на конечных пользователей

Основным направлением реализации разработанного продукта является применение его в качестве сервиса для обмена данными о местоположении между пользователями. Пользователем может быть любой человек, установивший разработанное мобильное приложение на свое мобильное устройство.

Продукт предназначен для массового использования в любом уголке земного шара, где есть доступ в Интернет.

ЗАКЛЮЧЕНИЕ

В результате выполнения магистерской диссертации разработано мобильное приложение для устройств, работающих на платформе iOS. Разработанное мобильное приложение выполнено в соответствии с требованиями и позволяет пользователям обмениваться данными о местоположении в режиме реального времени. Точность определения местоположения установлена в соответствии с требованиями. Реализован базовый инструментарий, такой как отображение пользователей на карте в режиме реального времени, создание и редактирование профиля и групп пользователей.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Охрана труда. Основы безопасности жизнедеятельности // www.Grandars.ru. URL: http://www.grandars.ru/shkola/bezopasnost-zhiznedeyatelnosti/ohrana-truda.html (дата обращения: 11.03.2017).
2. Воздействие шума на человека // GardenWeb. URL: http://gardenweb.ru/vozdeistvie-shuma-na-cheloveka (дата обращения: 10.03.2017).
3. СН 2.2.4/2.1.8.562-96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки // Библиотека гостов и нормативов. 2016. URL: http://ohranatruda.ru/ot_biblio/normativ/data_normativ/5/5212/ (дата обращения: 11.03.2017).
4. Попов В.М. Психология безопасности профессиональной деятельности: учебное пособие / В. М. Попов; Новосибирский государственный технический университет. – Новосибирск: Изд-во Новосибирского государственного технического университета, 1996 г. – 155 с.
5. СанПиН 2.2.2/2.4.1340-03. Санитарно-эпидемиологические правила и нормы. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы // Электронный фонд правовой и нормативно-технической документации. URL: http://docs.cntd.ru/document/901865498 (дата обращения: 10.03.2017).
6. ГОСТ Р 12.1.019-2009 ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты // Электронный фонд правовой и нормативно-технической документации. URL: http://docs.cntd.ru/document/1200080203 (дата обращения: 11.03.2017).
7. Чрезвычайные ситуации при работе с ПЭВМ // Студопедия — Ваша школопедия. URL: http://studopedia.ru/8_107307_osveshchenie-pomeshcheniy-vichislitelnih-tsentrov.html (дата обращения: 10.03.2017).

8. Долин П.А. Справочник по технике безопасности. М.: Энергоатомиздат, 1984 г. – 824 с.
9. Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 3.07.2016) // Электронный фонд правовой и нормативно-технической документации. URL: http://docs.cntd.ru/document/901807664 (дата обращения: 11.03.2017).
10. ГОСТ Р 50923-96 Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения // Электронный фонд правовой и нормативно-технической документации. URL: http://docs.cntd.ru/document/1200025975 (дата обращения: 11.03.2017).
11. ГОСТ 22269-76 Система "Человек-машина". Рабочее место оператора. Взаимное расположение элементов рабочего места. Общие эргономические требования // Электронный фонд правовой и нормативно-технической документации. URL: http://docs.cntd.ru/document/1200012834 (дата обращения: 11.03.2017).
12. ГОСТ 12.2.032-78 ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования // Электронный фонд правовой и нормативно-технической документации. URL: http://docs.cntd.ru/document/1200003913 (дата обращения: 11.03.2017).
13. Федеральный закон от 22.07.2008 г. №123 – ФЗ. "Технический регламент о требованиях пожарной безопасности".
14. Постановление Правительства РФ от 03.09.2010 N 681 (ред. от 01.10.2013) "Об утверждении Правил обращения с отходами производства и потребления в части осветительных устройств, электрических ламп, ненадлежащие сбор, накопление, использование, обезвреживание, транспортирование и размещение которых может повлечь причинение вреда жизни, здоровью граждан, вреда животным, растениям и окружающей среде // Консультант Плюс. 2015. URL: http://www.consultant.ru/document/cons_doc_LAW_104420/e1b31c36ed10

83efeb6cd9c63ed12f99e2ca77ed/#dst100007 (дата обращения: 03.04.2017).

15. Как утилизировать люминесцентную лампу? http://eco63.ru/lampalum.html (дата обращения: 03.04.2017).
16. iOS Architecture Patterns [Электронный ресурс]. – Режим доступа: https://medium.com/ios-os-x-development/ios-architecture-patterns-ecba4c38de52 - 2015. – Дата обращения: 25.02.2017.
17. Простым языком об HTTP [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/215117/. – Дата обращения: 15.03.2017.
18. HTTP запрос [Электронный ресурс]. – Режим доступа: http://citforum.ru/internet/cgi_tut/rqst.shtml. – Дата обращения: 15.03.2017.
19. Введение в JSON [Электронный ресурс]. – Режим доступа: http://www.json.org/json-ru.html. – Дата обращения: 18.03.2017.
20. Networking tutorial for iOS: How to create a socket based iPhone app and server [Электронный ресурс]. – Режим доступа: https://www.raywenderlich.com/3932/networking-tutorial-for-ios-how-to-create-a-socket-based-iphone-app-and-server. – Дата обращения: 25.04.2017.
21. Using sockets and socket streams [Электронный ресурс]. – Режим доступа: https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/NetworkingTopics/Articles/UsingSocketsandSocketStreams.html. Дата обращения: 25.04.2017.
22. Socket.IO [Электронный ресурс]. – Режим доступа: https://socket.io/blog/socket-io-on-ios/. – Дата обращения: 25.04.2017.

23. OMGHTTPURLQ [Электронный ресурс]. – Режим доступа: https://github.com/mxcl/OMGHTTPURLRQ. – Дата обращения: 19.03.2017.
24. Socket.io-client-swift [Электронный ресурс]. – Режим доступа: https://github.com/socketio/socket.io-client-swift. – Дата обращения: 25.04.2017.
25. Promises for Swift & ObjC [Электронный ресурс]. – Режим доступа: https://github.com/mxcl/PromiseKit. – Дата обращения: 19.03.2017.
26. Simple JSON Object mapping written in Swift [Электронный ресурс]. – Режим доступа: https://github.com/Hearst-DD/ObjectMapper. – Дата обращения: 19.03.2017.
27. Mode-View-Controller [Электронный ресурс]. – Режим доступа: https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html. – Дата обращения: 12.03.2017.

ПРИЛОЖЕНИЕ А

Раздел 1

Requirements analysis and design

Раздел 2

Realization

Студент:

Группа	ФИО	Подпись	Дата
8ИМ5А	Галузо Кирилл Борисович		

Консультант кафедры ИСТ

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент каф. ИСТ	Мирошниченко Е.А.	к.т.н.		

Консультант – лингвист кафедры ИЯИК

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель каф. ИЯИК	Горбатова Т.Н.	-		

1. REQUIREMENTS ANALASYS AND DESIGN

Requirements are based on given tasks and design of user interface. These requirements are presented as use cases.

1.1. Use cases

1.1.1. App states use case

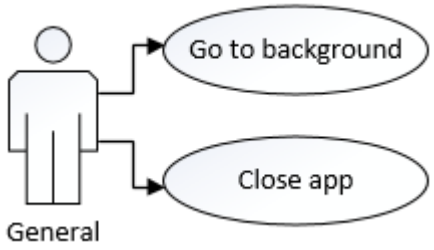

Figure 1.1 App state use case diagram

1.1.1.1. Use case «Go to background»

Goal: change app state.

Default state: app opened in foreground.

Scenario:

c. user moves app to background;

d. system changes method of determining user location to less sensitive and continues sending user location to the server every 10 meters.

1.1.1.2. Use case «Close app»

Goal: change app state.

Default state: app opened in foreground or background.

Scenario:

c. user removes app from list;

d. system changes method of determining user location to less sensitive and continues sending user location to server every 100 meter.

1.1.2. Sign In/Up use case

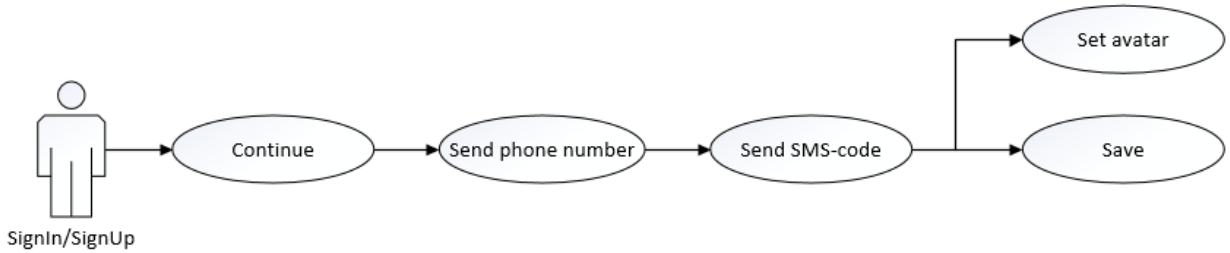

Figure 1.2 – Sign In/Up use case

1.1.2.1. Use case «Continue»

Goal: show phone confirmation screen.

Default state: user entered phone number.

Scenario:

c. user taps button «Continue»;

d. system shows phone confirmation screen.

1.1.2.2. Use case «Send phone number»

Goal: confirm entered before phone number and send it to the server.

Default state: user entered phone number and pressed «Continue» button.

Scenario:

c. user taps on button «Confirm»;

d. system sends phone number to the server:

- in case of success system saves phone number and opens «Enter SMS-code» screen;
- in case of failure system shows error.

1.1.2.3. Use case «Send SMS-code»

Goal: enter received SMS-code and send it to the server for confirming phone number.

Default state: received SMS-code.

Scenario:

d. user enters received SMS-code;

e. user taps «Continue» button;

f. system sends saved phone number and SMS-code to the server:

- in case of success system opens:
  - «Create profile» screen in case of sign up;
  - «Map» in case of sign in.
- in case of failure system shows error.

1.1.2.4. Use case «Set avatar»

Goal: set profile avatar (optional).

Default state: profile created.

Scenario:

e. user taps on «Avatar» button;

f. system shows context default iOS context menu for picking image.

g. user picks image;

h. system sends image to the server:

- in case of success system shows uploaded avatar;
- in case of failure system shows error.

1.1.2.5. Use case «Save»

Goal: save changes to new profile.

Default state: screen «Create profile» opened, user set avatar and entered name (both are not necessary to finish registration procedure).

Scenario:

c. user taps «Save» button;

d. system sends modified profile to the server:

- in case of success system opens «Map» screen;
- in case of failure system shows error.

1.1.3. Main screen use case

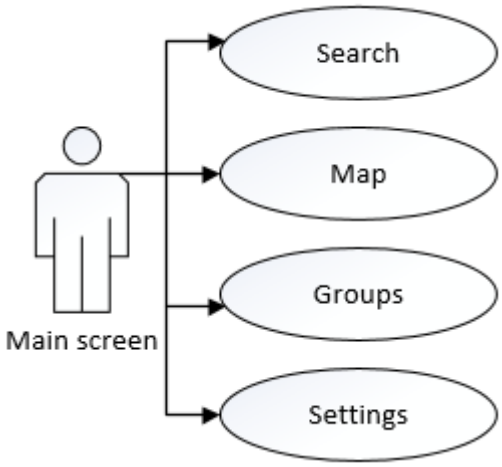

Figure 1.3 – Main screen use case diagram

1.1.3.1. Use case «Search»

Goal: open «Search» screen

Default state: opened «Map», «Groups» or «Settings» screen.

Scenario:

c. user taps «Search» button on tab bar;

d. system opens «Search» screen with last search result.

1.1.3.2. Use case «Map»

Goal: open «Map» screen.

Default state: opened «Search», «Groups» or «Settings» screen.

Scenario:

c. user taps «Map» button on tab bar;

d. system opens «Map» screen with map centered at user’s location.

1.1.3.3. Use case «Groups»

Goal: open «Groups» screen.

Default state: opened «», «» or «» screens.

Scenario:

c. user taps «Groups» button on tab bar;

d. system opens «Groups» screen.

1.1.3.4. Use case «Settings»

Goal: open «Settings» screen.

Default state: opened «Search», «Map» or «Groups» screens.

Scenario:

c. user taps «Settings» on tab bar;

d. system opens «Setting» screen.

1.1.4. Search use case

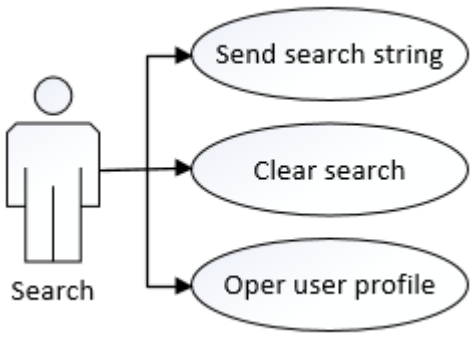

Figure 1.4 – Search use case diagram

1.1.4.1. Use case «Send search string»

Goal: search for user(s).

Default state: opened «Search» screen, user entered search string.

Scenario:

c. user taps «Search» button;

d. system sends search string to the server:

- in case of success system reloads search results;
- in case of failure system shows error.

1.1.4.2. Use case «Clear search»

Goal: clear search text field and search results.

Default state: user entered search string in search text field.

Scenario:

c. user taps «Clear» button;

d. system clears search string in search text field and removes search results.

1.1.4.3. Use case «Open user profile»

Goal: open «Profile» screen with selected user profile data.

Default state: search succeed, search results loaded.

Scenario:

c. user taps cell in search result table;

d. system opens «Profile» screen with tapped user profile data.

1.1.5. Map use case

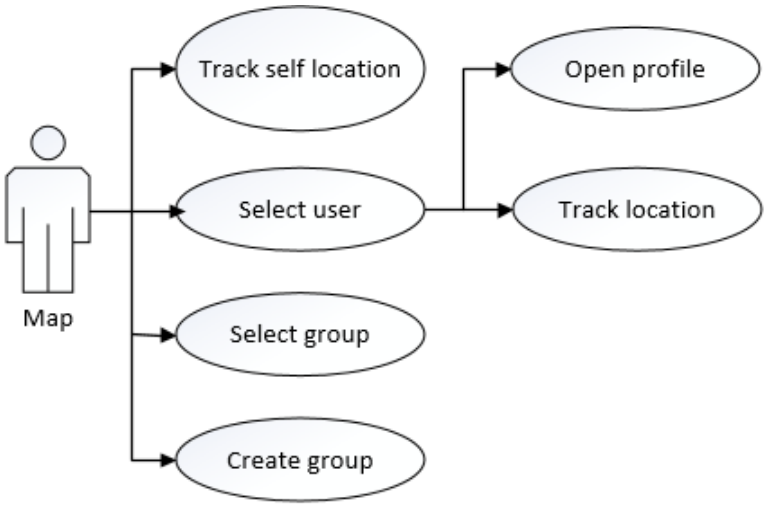

Figure 1.5 – Map use case diagram

1.1.5.1. Use case «Track self location»

Goal: track self location on map.

Default state: «Map» screen opened.

Scenario:

c. user taps «Track me» button;

d. system starts following users location on the map.

1.1.5.2. Use case «Select user»

Goal: show user's profile info on the map.

Default state: «Map» screen opened, other user's annotation loaded on the map.

Scenario:

d. user taps other user annotation on the map;

e. system shows callout view for tapped annotation filled with user info;

f. System shows «Track user» button.

1.1.5.3. Use case «Open profile»

Goal: open tapped user profile.

Default state: «Map» screen opened, other user annotation tapped, callout view for tapped user showed.

Scenario:

c. user taps callout view for selected user;

d. system opens «Profile» screen with tapped user profile data.

1.1.5.4. Use case «Track tapped user's location»

Goal: track tapped user location on the map.

Default state: «Map» screen opened, other user annotation tapped.

Scenario:

c. user taps «Track user» button;

d. system follows tapped user location on the map.

1.1.5.5. Use case «Select group»

Goal: show location of users who are members of tapped group.

Default state: «Map» screen opened.

Scenario:

c. user taps group in the section above the map;

d. system removes all user's annotation from map and load annotations for locations of members of tapped group.

1.1.5.6. Use case «Create group»

Goal: create new group of users.

Default state: «Map» screen opened.

Scenario:

c. user taps «Add group» button in section above the map;

d. system opens «Create group» screen.

1.1.6. Groups use case

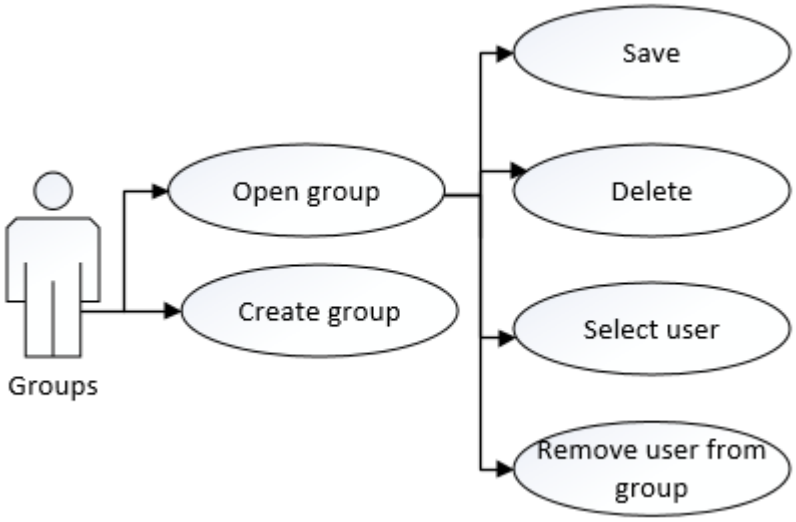

Figure 1.6 – Groups use case diagram

1.1.6.1. Use case «Open group»

Goal: show screen with group info (group name, group visibility, group members)

Default state: «Groups» screen opened.

Scenario:

c. user taps group cell in group collection

d. system loads group info:

- in case of success system opens «Group» screen with filled group info
- in case of failure system shows error.

1.1.6.2. Use case «Delete»

Goal: delete group.

Default state: «Group» screen opened, loaded group is not a one of standard groups (Friends, Family or Blacklist).

Scenario:

c. user taps «Delete group» button;

d. system sends delete request to the server:

- in case of success system opens «Groups» screen with reloaded group collection;
- in case of failure system shows error.

1.1.6.3. Use case «Save»

Goal: save changes.

Default state: «Group» screen opened.

Scenario:

c. user removes group member(s), changes group name or group visibility, taps «Save» button;

d. system sends modified group to the server:

- in case of success system opens «Groups» screen with reloaded group collection;
- in case of failure system shows error.

1.1.6.4. Use case «Select user»

Goal: open «Profile» screen with loaded tapped user profile info.

Default state: «Group» screen opened, group member(s) showed in group members table.

Scenario:

c. user taps user in group members table;

d. system opens «Profile» screen with tapped user profile data.

1.1.6.5. Use case «Remove user from group»

Goal: remove user from group.

Default state: «Group» screen opened, group member(s) showed in group members table.

Scenario:

c. user taps «Delete user» button in group members table cell;

d. system removes appropriate cell from group members table.

1.1.7. Create group use case

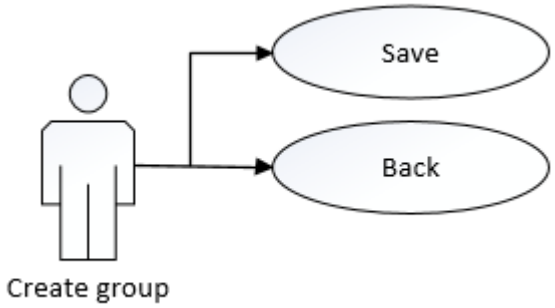

Figure 1.7 – Create group use case diagram

1.1.7.1. Use case «Save»

Goal: save new group.

Default state: «Create group» screen opened, group name entered in «Group name» text field.

Scenario:

c. user taps «Save» button;

d. system sends new group to the server:

- in case of success system opens «Groups» screen with reloaded groups, if group creation initialized on it, or «Map» screen with reloaded group collection, if group creation initialized on it;
- in case of failure system shows error.

1.1.7.2. Use case «Back»

Goal: cancel creation of new group and open previous screen.

Default state: «Create group» screen loaded.

Scenario:

c. user taps «Back» button;

d. system closes «Create group» screen and opens previous screen.

1.1.8. Settings use case

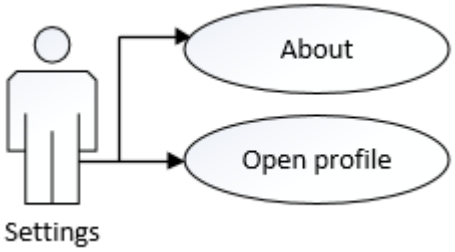

Figure 1.8 – Settings use case diagram

1.1.8.1. Use case «About»

Goal: open «About» screen with info about application.

Default state: «Settings» screen opened.

Scenario:

c. user taps «About» button

d. system opens «About» screen with info about application.

1.1.8.2. Use case «Open profile»

Goal: open «Profile» filled with user’s profile data.

Default state: «Setting» screen opened.

Scenario:

c. user taps «Profile» button;

d. system opens «Profile» screen filled with user’s profile data.

1.1.9. Profile use case

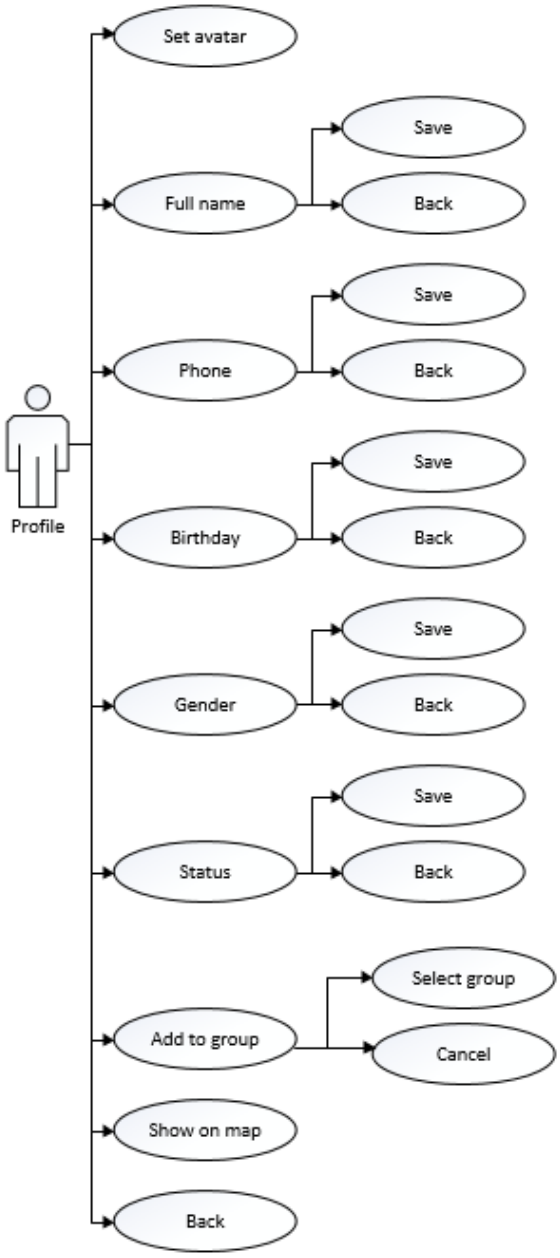

Figure 1.9 – Profile use case diagram

1.1.9.1. Use case «Set avatar»

Goal: set profile avatar.

Default state: profile created.

Scenario:

i. user taps on «Avatar» button;

j. system shows context default iOS context menu for picking image.

k. user picks image;

l. system sends image to the server:

- in case of success system shows uploaded avatar;
- in case of failure system shows error

1.1.9.2. Use case «Full name»

Goal: open «Full name» screen.

Default state: «Profile» screen opened.

Scenario:

c. user taps «Full name» cell;

d. system opens «Full name» screen.

1.1.9.3. Use case «Phone»

Goal: open «Phone» screen.

Default state: «Profile» screen opened.

Scenario:

c. user taps «Phone» cell;

d. system opens «Phone» screen.

1.1.9.4. Use case «Birthday»

Goal: open «Birthday» screen.

Default state: «Profile» screen opened.

Scenario:

c. user taps «Birthday» cell;

d. system opens «Birthday» screen.

1.1.9.5. Use case «Gender»

Goal: open «Gender» screen.

Default state: «Profile» screen opened.

Scenario:

c. user taps gender cell;

d. system opens «Gender» screen.

1.1.9.6. Use case «Add to group»

Goal: add user to group.

Default state: «Profile» screen opened, loaded other user’s profile.

Scenario:

e. user taps «Add to group» button;

f. system opens «Add to group» screen with loaded group collection;

g. user picks group;

h. system sends request to the server to add user to selected group:

- in case of success system closes current screen and opens «Profile» screen;
- in case of failure system shows error.

1.1.9.7. Use case «Show on map»

Goal: show other user on the map.

Default state: «Profile» screen opened, loaded other user's profile.

Scenario:

c. user taps «Show on map» button;

d. system opens «Map» screen with map centered on other user's location.

1.1.9.8. Use case «Save»

Goal: save edited profile field.

Default state: «Full name», «Phone», «Gender», «Birthday» or «Status» screen opened, edited profile field or profile filed visibility.

Scenario:

c. user taps «Save» button;

d. system sends request to server:

- in case of success system closes current screen and opens «Profile» screen with reloaded profile;
- in case of failure system shows error.

1.1.9.9. Use case «Back»

Goal: close current screen, discard all changes and open previous screen.

Default state: «Full name», «Phone», «Gender», «Birthday» or «Status» screen opened.

Scenario:

c. user taps «Back» button;

d. system closes current screen and opens previous.

2. REALIZATION

2.1. Architecture

This project includes two applications: mobile application for iOS devices and server application.

Server application handles location data and saves this data to data base. Also server application stores profiles data which are stored in data base too. Amazon AWS service is being used to store uploaded user photos.

Mobile application sends location data, sends and receives profile, groups data by using server's API. Mobile app receives other users location data from the server by using web-sockets.

Architecture of the project is presented in figure 2.1.

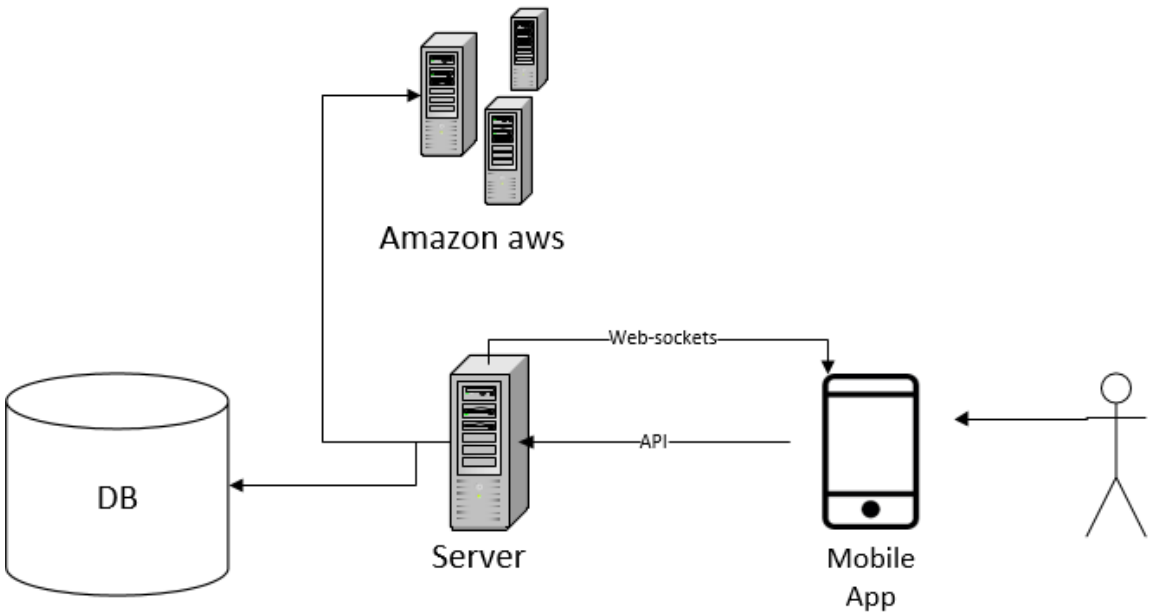

Figure 2.1 – project architecture

Architecture of mobile application is based on VIPER architecture. It is a popular architecture pattern. The scheme is presented in figure 2.2.

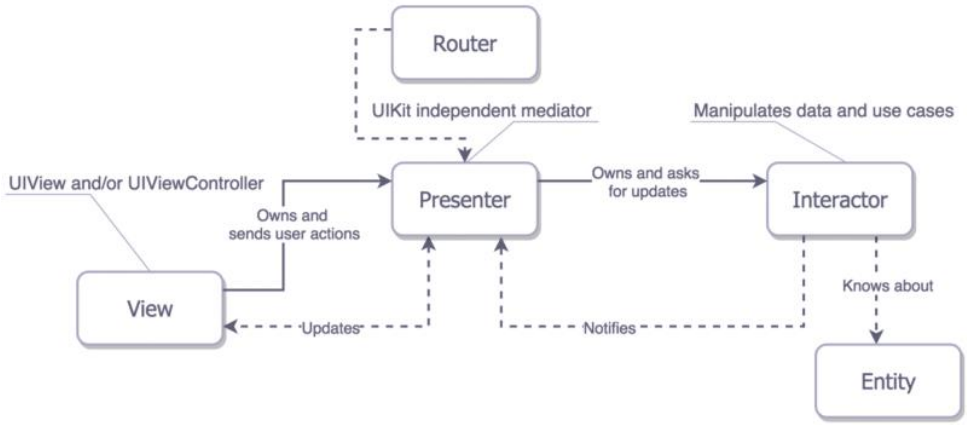

Figure 2.2 – VIPER scheme

VIPER module can be presented as one screen or user story (for example: the process of authentication can be in one or more connected screens). Realization of VIPER module is presented in figure 2.3.

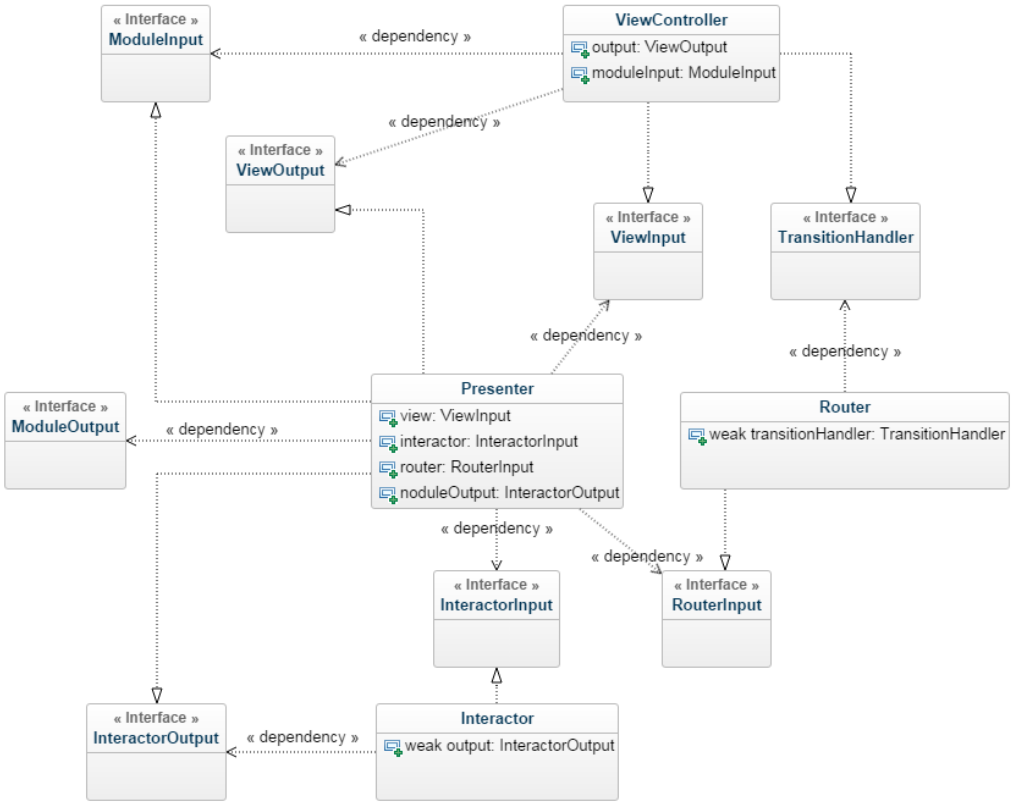

Figure 2.3 – Realization of VIPER module

Presenter is the center of VIPER module. Presenter class is implementation of protocol. Description of each element of module is presented in table 1.

Table 1 – VIPER module elements description

View controller	Handles all user's actions and updates User Interface (UI)
View Input	Interface for access for View Controller from Presenter
View Output	Interface for access Presenter from View Controller
Presenter	Responsible for module logic
Module Input	Interface for access from outside the module
Module Output	Interface for access other module which implements this interface
Router	Responsible for switching between modules
Router Input	Interface for access Router from Presenter
Transition Handler	Interface for access View Controller from Router for switching between modules
Interactor	Responsible for business logic and has access for Business layer
Interactor Input	Interface for access Interactor from Presenter
Interactor Output	Interface for access Presenter from Interactor

In comparison with MV(X) patterns VIPER has several differences:

- logic moved to Interactor from Model, Entities are just a data structures;
- responsibilities for UI presentation moved from Controller, Presenter, View Model to Presenter without ability of changing data;
- VIPER has the best in distribution of responsibilities between module elements;
- testability is better because of the best distribution of responsibilities.

2.2. Network libraries

OMGHTTPURL, PromiseKit, Soket.IO and ObjectMapper were used for network processes in the project.

2.2.1. OMGHTTPURL

OMGTTPURL is an extension for Apple standard NSURLRequest protocol which encapsulates two basic elements of load request: the URL to load, and policy to use when consulting the URL content cache made available by implementation.

NSURLRequest was designed to be extendable to support additional protocols by adding categories that provide accessory methods for own protocol-specific properties.

OMGHTTPURL is extension for NSURLRequest. It wraps requests configuration in simple methods which are used by developers. It makes code cleaner and more readable. Developers do not need to write a lot of lines of code for configuring requests; they need only specify necessary properties in methods.

For example, you need to write these lines to create POST-request with given URL and JSON:

```
let request = NSMUtableRequest()
request = OMGHTTPURLQ.post(urlString, json: json)
```

OMGHTTPURL is a part of PromiseKit.

2.2.2. PromiseKit

PromiseKit provides Promises which are very useful as asynchronous tasks they represent. PromiseKit has converted almost all of Apple's APIs to Promises.

All networking tasks are asynchronous so PromiseKit is perfectly suitable for these tasks.

For sending and handling requests in project I used services and repositories.

If I need to send request to the server I call appropriate service method which calls repository method witch calls methods for creating and sending request.

As VIPER architecture says, Intearctor is responsible for business logic, so calling service methods are in Interactor.

```
import PromiseKit
class ExampleInteractor: ExampleInteractorInput {
      weak var output: ExampleInteractorOutput?
      let exampleService = ExampleServiceType
      init(exampleService: ExampleServiceType) {
           self.exampleService = exampleService
      }
```

```
          func sendSomeString(exampleString: String) {
               firstly {
                    exampleService.sendString(exampleString:
exampleString)
               }.then { response in
                    self.handle(response)
               }.catch { error in
                    self.output?.showError(error.message())
               }.always {
                    self.alwaysExampleMethod()
               }
          }
     }
```

Method sendSomeString has String object as an input parameter. ExapmpleServiceType protocol has sendString method. Object named exampleService is an object of class encapsulating ExampleServiceType protocol. Method named sendString is a method returning URLDataPromise or just Promise. There is a chain of Promises which starts at the moment of sending request to the server and ends at Interacor. If error occurs somewhere in this chain it throws directly to the end of chain.

Block «then» is called when chain of promises returns response. If exampleService method returns some value this value should be handled inside Interactor, and only then it should notify Presenter.

Block «catch» is called when error occurs somewhere in chain of promises. Presenter implements ExampleInteractorOutput protocol, so if there is some error Interactor notifies Presenter through this protocol about error.

Block «always» is called when Promises return. There can be Interactor's method which do some data preparations and then notify Presenter, or there can be direct call self.output?.someMethod(), if there is no need in data preparation.

2.2.3. Soket.IO

Socket.IO-ios-swift is socket client for using socket in iOS application written in swift. It uses websockets to communicate with server.

The websocket communication relies on the client-server logic, where connection between client and server always exists.

In this project sockets are being used for updating users location on the map in real-time.

Usage of this socket client is very simple.

First of all you need to create socket client.

```
let socket = SocketIOClient(socketURL: "youtServerURLString:8900")
```

Then you have to determine event handlers.

```
socket.on("connect") { data, ack in
      self.socket.emit("listen user", with [token])
}
socket.on("message") { data, ack in
      //notify delegate
}
```

Method «on» is called by socket client when receives message from server with appropriate tag. For example, when initializing connection between client and server, client receives message «connect» from server and sends message «listen user» with token object to server. Then server starts listening for events from this client and sending to it messages if there are any.

Method «emit» sends message to server throw sockets.

To establish connection between server and client «connect» method should be called.

```
func startListeningForEvents() {
      socket.on("connect") {…}
      socket.on("message") {…}
      socket.connect()
}
```

It is nessecary to call «disconnect» method to stop connection between server and client.

```
func stopListening() {
      socket.disconnect()
}
```

2.2.4. ObjectMapper

ObjectMapper is a framework written in Swift in order to transform a model to JSON and vice versa.

When data is received from server it has to be converted to model object for further use in application.

Structs are preferable to be used as data model objects, because they are relatively small and can be copied. Copying is way safer than having multiple reference to the same instance as happens with classes. This is especially important when passing around a variable to many classes and/or in a multithreaded environment. If you send a copy of your variable to other places, you never have to worry that other part of your code change the value of your variable.

With structs there is much less need to worry about memory leaks or multiple threads racing to access/modify a single instance of a variable.

Mappable or ImmutableMappable protocol of ObjectMapper framework should be implemented in order to convert JSON to data model structure.

Protocol Mappable is used in general. But ImmutableMappable can be used when you are sure that one of model properties will be received from server with 100% chance.

```
struct User{
      var id: Int
      var firstName: String?
      var lastName: String?
      var photo: Photo?
}
extension User: ImmutableMappable {
      init(map: Map) throws {
            id = try map.value("userId")
      }
      mutating func mapping(map: Map) {
            id        <- map["userId"]
            firstName <- map["first_name"]
            lastName  <- map["last_name"]
            photo          ,_ map["photo"]
      }
}
```

This is example of data structure that will be converted from JSON. This process is called mapping.

First of all, we declare properties of structure User such as id, first and last names and photo. Properties can be different type, even other data structures. Swift grants ability to use Optional types (String?). Properties of these types can have value or can be nil. Optional object should be unwrapped before usage.

```
    var optionalValue: String? = “some string”
    var someString = “optional string is going to be after this
text” + optionalValue!
```

Optional properties of structure User are that properties that can be nil. User must have an «id», and there is 100% chance that server sends this value.

Next step is to extend structure User to implement ImmutableMappable protocol. In «init» method which can throw an error, if it occurs inside this method, should be initialized non-optional properties of structure.

Method «mapping» associates values from JSON to User structure properties by keys (userId, first_name, last_name, photo).

Mapping data with usage of PromiseKit:

```
class ExampleRepoSitory: ExampleRepositoryType {
      let mapping: ExampleMappingType
      init(mapping: ExampleMappingType) {
            self.mapping = mapping
      }
      func mapExample() {
            //send some GET request which returns data
            firstly {…
            }.then { data in
                  self.mapping.map(data: data)
            }
      }
}
```