МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(Национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

ПРОЕКТ ПРОВЕРЕН
Рецензент
Ведущий программист
_____Е.А. Жорницкий
«____» _____ 2020 г
ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.э.н. с.н.с.
_____Б.М. Суховилов
«____» _____ 2020 г

Мобильная версия сервиса EcoRadar.online для информирования пользователя об уровне загрязнении воздуха

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОМУ
КВАЛИФИКАЦИОННОМУ ПРОЕКТУ
ЮУрГУ– 09.03.02 2019.44 ПЗ ВКП

Консультанты:

Экономическая часть, старший
преподаватель
_____А.Г. Шепталин
«____» _____ 2020 г.

Техническая часть, старший
преподаватель
_____С.Г. Ботов
«___» _____ 2020 г.

Руководитель работы, доцент
_____С.А. Тимаева
«___» _____ 2020 г.

Автор проекта
студент группы ЭУ - 542
_____К.С. Терехов
«___» _____ 2020 г.

Нормоконтролёр, доцент
_____ С.А. Тимаева
«___» _____ 2020 г.

Челябинск 2020 г.

# АННОТАЦИЯ

Терехов К.С. «Мобильная версия сервиса EcoRadar.online для информирования пользователя об уровне загрязнении воздуха» – Челябинск: ЮУрГУ, ЭУ-542, 80 стр., 27 ил., 12 табл., 1 гр., библиогр. список – 35 наим.

Дипломный проект выполнен с целью оповещения населения города Челябинск об уровне загрязнении окружающей среды. Данные, которое будет использовать решение будут поступать от сервиса Ecoradar.online.

Рассмотрена миссия и стратегические цели организации. Выявлены основные пути достижения целей. Проанализировано дальнее и ближнее внешнее окружение предприятия, и его влияние на работу организации.

Исследованы методы разработки сервисов внутри компании «Infinnity». Сделаны выводы об организации её внутренних процессов. Исследована структура API, предоставляемого сервисом Ecoradar.online. Именно из этого API приложение получает необходимые данные. Среди получаемых сущностей – уровень загрязнения воздуха, количество содержащихся частиц и общие метеорологические данные.

В результате исследования был предложен проект разработки мобильного приложения, которое получает необходимые данные с датчиков и отображает их пользователю, таким образом, оповещая его об уровне загрязнения среды в той точке, в которой он находится сейчас. Приложение будет работать не только на основе текущих данных, но и использовать статистику. Таким образом, получая более точные данные.

В ходе проекта было составлено расписание проекта, определен перечень ресурсов и работ проекта, рассмотрены риски, которые могут возникнуть при внедрении. Была произведена оценка финансовой эффективности проекта.

**ОГЛАВЛЕНИЕ**

# ВВЕДЕНИЕ

В Челябинске в последние годы проблема загрязнения окружающей среды становится всё более и более актуальной. Несмотря на это, существует дефицит общедоступной информации на тему состояния экологии в городе. Есть возможность получить необработанные статистические данные, но не существует инструмента для отображения их в упорядоченном, доступном виде. Для решения это этой проблемы, необходимо реализовать приложение, которое будет доставлять информацию об актуальных данных касательно экологии – пользователям. Не менее важной является потребность получать данные как можно скорее, «в реальном времени».

Объект исследования: данные об экологической ситуации на территории города Челябинска.

Предмет исследования: данные, полученные с датчиков, отслеживающих экологические показатели в городе Челябинске, установленные в рамках проекта «Ecoradar.online».

Цель проекта: на основании данных, полученных с датчиков, реализовать приложение, позволяющее пользователям отслуживать состояние экологии в Челябинске.

Задачи выпускного квалификационного проекта: описать структуру и виды деятельности компании ООО «Infinnity Solutions», провести анализ ее ближнего и дальнего окружения, определить основные проблемы отображения данных с датчиков, предложить решение выявленной проблемы в рамках проекта «Ecoradar.online», рассмотреть способы усовершенствования способа отображения данных, разработать приложения для получения и отображения данных с датчиков «в реальном времени».

# ГЛАВА 1. ПРЕДПРОЕКТНОЕ ОБСЛЕДОВАНИЕ ПРЕДПРИЯТИЯ И СЕРВИСА

## 1.1. Информация о предприятии

2.1.1 Общие сведения

ООО «Infinnity Solutions» - отечественная компания-разработчик программного обеспечения, специализирующаяся на создании информационных систем и компонентов, бизнес веб-приложений и корпоративных решений, используемых в различных сферах деятельности: в коммерческой, производственной и социальной.

2.1.2 История

Компания «Infinnity Solutions» открылась в 2006 году. Для создания программных продуктов используется платформенное программное обеспечение собственной разработки. Благодаря развитому инструментарию платформы компания реализует как корпоративные проекты, так и собственные облачные SaaS решения. [1]

## 1.2. Характеристика

**1) Вид деятельности**

Основной вид деятельности компании «Infinnity Solutions» – заказная разработка программного обеспечения.

**2) Виды продукции**

Виды выпускаемого программного обеспечения:

- Web-приложения;
- Корпоративные приложения;
- Мобильные приложения;

**3) Виды услуг**

- Бизнес-анализ предметной области;
- Заказная разработка программного обеспечения;
- Внедрение ПО;

- Поддержка, сопровождение и обслуживание ПО.

### 1.3. Миссия и цели

1) **Миссия**

Миссия компании: предоставляем первоклассные решения в сфере информационных и коммуникационных технологий, будучи гибкой, сильной и надежной компанией и опираясь на профессиональные знания, команду и отношения с нашими заказчиками.

2) **Цели**

Стратегическая карта — это элемент системы сбалансированных показателей.

Представляет собой диаграмму, на которой обозначены основные цели существования организации. Цели на карте связаны между собой направленными причинно-следственными связями. Связи позволяют проследить воздействие одной цели на другую. Насколько достижение одной цели влияет на достижение связанной зависимой цели. Стратегические карты позволяют формализовать путь развития организации. Для понимая того, к какому виду внутренней деятельности относится та или иная цель предназначены перспективы.

Стандартный набор перспектив:

- "Обучение персонала и развитие";
- "Внутренние бизнес-процессы";
- "Клиенты и внешнее окружение";
- "Финансы".

Связь между целями показывает влияние одной перспективы на другую.

Стратегическая карта изображена на рисунке 1.

В качестве основной цели компании «Infinnity Solutions» была взята цель «Увеличение прибыли компании». Для достижения данной цели необходимо добиться увеличения доходов и снижения расходов. Для увеличения доходов можно использовать следующие способы: привлекать новых клиентов, повышать удовлетворенность клиентов предоставленными продукцией и услугами, удерживать старых клиентов и снижать время обслуживания клиентов.

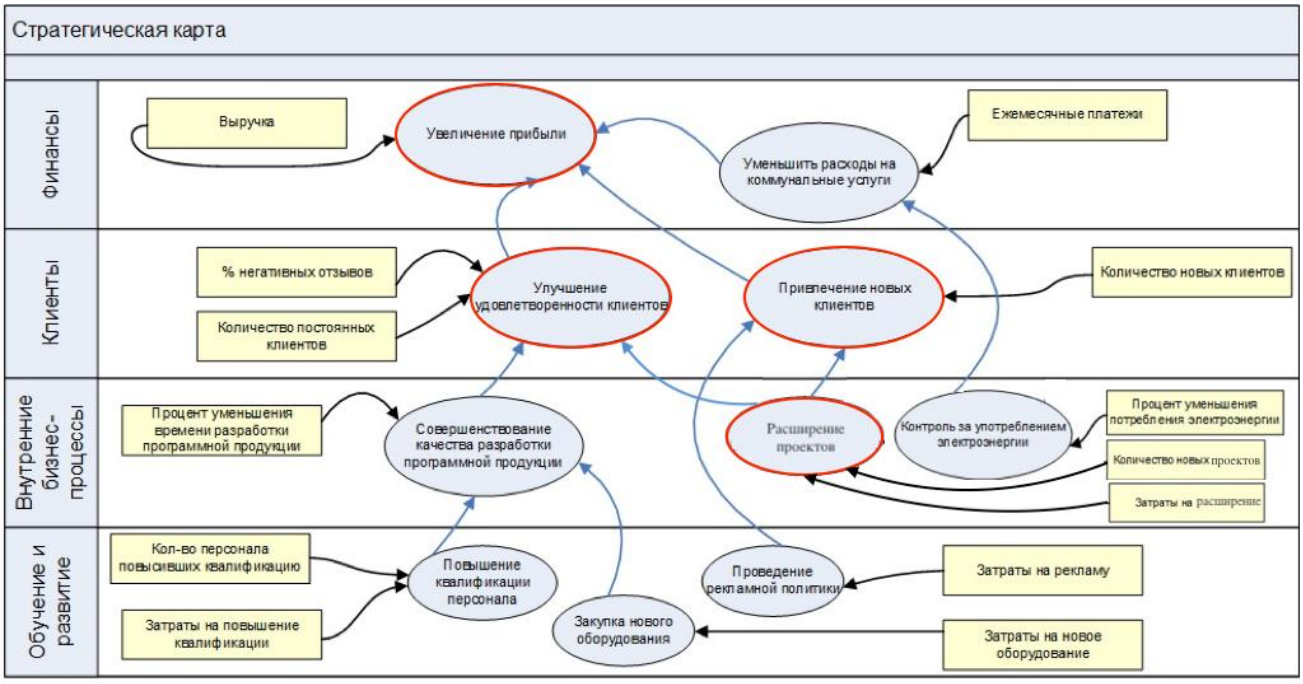


Рисунок 1 - Стратегическая карта целей

Привлечь новых клиентов можно расширением проектов и проведением рекламной компании. Также важно осознавать необходимость повышения квалификации сотрудников. Для снижения расходов требуется уменьшить расходы на коммунальные услуги, контролируя употребление электроэнергии, а также повысить удовлетворенность клиентов и совершенствования процесса разработки программной продукции, что предполагает замену устаревшего программного оборудования.

На стратегической карте устанавливаются целевые показатели, представляющие собой конкретные значения, которые стремится достичь организация, представленные в таблице 1.

Таблица 1 – Счетная карта

| Перспектива | Цель | Показатель | Значение |
|---|---|---|---|
| | | | Цели к 2023 году |
| Финансы | Рост прибыли | Выручка | +20% |
| | Уменьшить расходы за коммунальные услуги | Ежемесячные платежи | -10% |
| Клиенты | Улучшение удовлетворенности клиентов | Процент негативных отзывов | 0% |
| | | Количество постоянных клиентов | +20% |

Продолжение таблицы 1

| | | | |
|---|---|---|---|
| | Увеличить количество клиентов | Количество новых клиентов | +10% |
| Бизнес-процессы | Совершенствование качества разработки программной продукции | Процент уменьшения времени разработки программных продуктов | -10% |
| | Контроль за употреблением электроэнергии | Процент уменьшения потребления электроэнергии | 20% |
| | Расширение проектов | Количество новых проектов | +10% |
| | | Затраты на расширение | ≤600 000 руб. |
| Повышение узнаваемости компании | Увеличение количества упоминаний в СМИ | Количество упоминаний в региональных СМИ | +10 |
| | | Количество упоминаний в федеральных СМИ | +5 |
| | Увеличение количества скачиваний приложений | Количество скачиваний приложений | +20 000 |
| | Увеличение количества положительных отзывов в социальных сетях | Количество положительных отзывов в социальных сетях | +50 |

## 1.4. Функциональная структура предприятия

Функциональная структура организации представлена на рисунке 2.

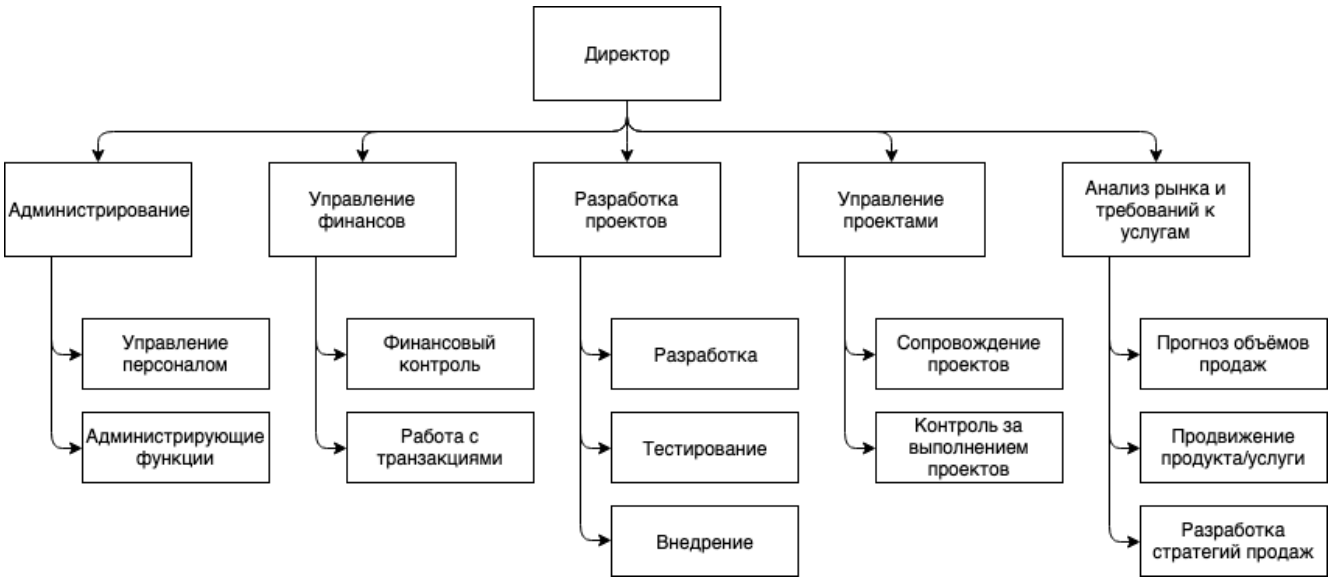


Рисунок 2 -Функциональная структура компании

### 1.5. Организационная структура предприятия

Генеральный директор компании – Гребнев Павел Евгеньевич.

За проектирование программного обеспечения отвечает отдел аналитики, за разработку – отдел разработки, за продвижение – единая служба сопровождения проектов. Продвижением продукции и услуг компании занимаются сотрудники отдела маркетинга. За набор персонала, адаптацию сотрудников к специфике деятельности компании отвечают руководители отделов и исполнительный директор.

Организационная структура компании представлена на рисунке 3.

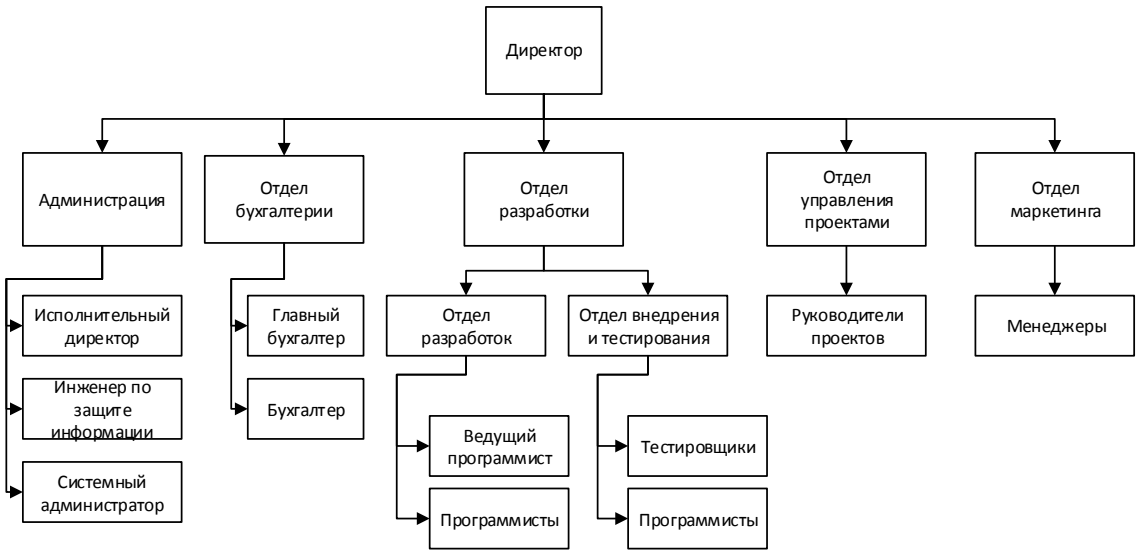


Рисунок 3 -Организационная структура компании

Вывод: в компании «Infinnity Solutions» используются развитые функциональная и организационная структуры, которые отвечают современным требованиям и справляются со всеми поставленными задачами.

### 1.6. Анализ внешней среды

## STEEP- анализ дальнего окружения

Макросреда – это процессы и явления государства, на которые система повлиять не может, но которые влияют на долгосрочные решения системы. [2] STEEP - анализ рассматривает такие факторы, как:

1. S – социальные
2. T – технологические
3. E – экономические
4. E – экологические
5. P – политические

**Цель -** выявить угрозы и положительные факторы для определения стратегии дальнейшего функционирования компании под действием внешней среды**.**

Социальные (S):

Увеличение потребности на использование приложений для мобильных устройств, связанное с появлением у людей необходимости в своевременном получении информации разных видов и различной тематики, вероятно, приведёт к повышению спроса на программную продукцию компании, росту объема продаж и получаемой прибыли.

Экологическая ситуация в регионах России стабильно отстаёт от аналогичной во многих странах запада. Промышленных городов, таких как Челябинск это касается в первую очередь. В 2018 году в Челябинске было несколько скандалов, связанных с экологией. Это и смог на улицах города, и выбросы, о которых СМИ сообщили с огромной задержкой, и «мусорных коллапс» когда несколько дней мусор не вывозился из города из-за конфликта ответственных за это компаний. Социальный запрос на изменение экологической ситуации велик [1]

Технологические (Т):

Развитие технологий, позволяющее создавать мульти-платформенные приложения или прогрессивные web-приложения, позволит компании «Infinnity Solutions» в более сжатые сроки создавать приложения более высокого качества. Таким образом, компания может увеличить лояльность клиентов и вызвать рост количества продаж. Это, в свою очередь, приведёт к общему росту прибыли компании.

Если конкуренты компании «Infinnity Solutions» выпустят более технически совершенное программное обеспечение, это может привести к тому, что конкурентоспособность компании станет ниже. Так же, соответственно, снизится объём продаж и общая прибыль.

Экономические (Е):

Спад цен на необходимое оборудование и программное обеспечение поможет компании в обновлении аппаратного обеспечения и программных инструментов. Это приведёт к удешевлению и увеличению скорости и производительности разработки программных продуктов. Это улучшит надёжность разрабатываемого программного обеспечения, а также его качество в целом, что повысит конкурентоспособность фирмы на рынке и увеличит объём получаемой ей прибыли.

Понижение спроса на программные продукты компании «Infinnity Solutions», связанные с финансовым кризисом, высокой стоимостью товара или избытком решений такого рода на рынке, могут привести к существенному падению объёма продаж и прибыли.

Экологические (Е):

Увеличение контроля за экологией приводит к необходимости автоматизации некоторых аспектов этого контроля.

Это может привести к новым заказам компании и увеличению прибыли.

Политические (Р):

Дополнительные налоговые сборы за разработку программного обеспечения могут привести к падению прибыли компании и к снижению рентабельности разработки программного обеспечения. Новые запреты могут сделать разработку некоторых программных продуктов невозможной.

Поддержка отечественных разработчиков и снижение экспортных пошлин позволит компании укрепить свои позиции не только на рынках родной страны, но и на зарубежных. Это позволит компании увеличить свой доход и даст дополнительную прибыль.

Возможный сценарий развития с учётом вышеперечисленных факторов представлен в таблице 2 и на рисунке 4.

Таблица 2 - Профиль макросферы

| Факторы | Знак | Качественная оценка | Балл | Вес | Важность | Критический синтез |
|---|---|---|---|---|---|---|
| **Социальные** | | | | | | |
| 1.Увеличение спроса на мобильные технологии | + | сильное | 8 | 0,15 | +1,2 | Оценка затрат и мониторинг рынка для перехода на новые технологии |
| 2.Осведомлённость об экологической ситуации | - | существенное | 5 | 0,06 | -0,3 | Привлечение пользователей |
| **Технологические** | | | | | | |
| 3.Развитие мультиплатформенных технологий | + | существенное | 4 | 0,05 | +0,2 | Оценка затрат и мониторинг рынка для перехода на новые технологии и привлечение клиентов |
| 4.Выпуск инновационных продуктов другими компаниями | - | существенное | 5 | 0,06 | -0,3 | Ускорение производительности |
| **Экономические** | | | | | | |
| 5.Снижение стоимости компьютерного | + | значительное | 6 | 0,8 | +0,48 | Замена оборудования |

| | | | | | | |
|---|---|---|---|---|---|---|
| оборудования | | | | | | |
| 6.Экономический кризис | - | значительное | 7 | 0,1 | -0,7 | Сохранение денежных средств |
| **Экологические** | | | | | | |
| 7.Увеличение контроля за экологией | + | слабое | 4 | 0,05 | +0,2 | Сохранение денежных средств |
| **Политические** | | | | | | |
| 8.Ввод санкций, действующий на IT-отрасль | + | значительное | 8 | 0,15 | +1,2 | Сохранение денежных средств |
| 9.Ужесточение требований разработки программного приложения | - | существенное | 8 | 0,15 | -1,2 | Переход на удобную систему налогового учета |
| 10.Ужесточение закона «О персональных данных» | - | существенное | 8 | 0,15 | -1,2 | Сохранение денежных средств, привлечение клиентов |
| Итого: | | | | 1 | -0,42 | |

**Вывод:** нужно уделить внимание факторам: экономический кризис, ужесточение требований разработки программного обеспечения и ужесточение закона «О персональных данных», так как они оказывают наиболее отрицательное влияние на организацию. Однако, положительные факторы (рост спроса на технологии и ввод санкций, действующий на IT-отрасль) могут помочь компании в преодолении трудностей.

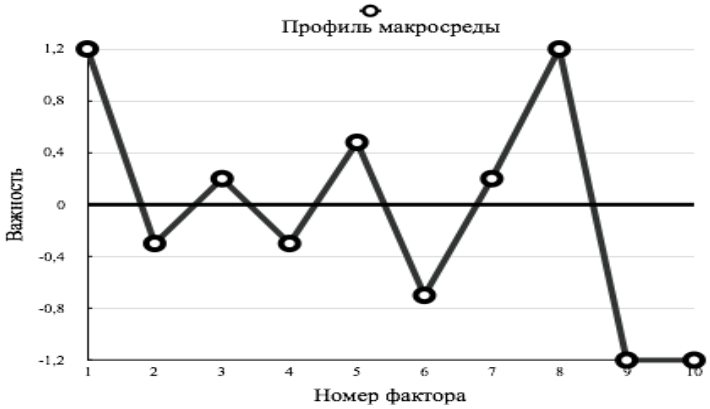


Рисунок 4-Профиль «Макросреды»

Для компании макросреда является отрицательной - суммарный показатель важности -0,42. Таким образом, компания «Infinnity Solutions» находится в негативной внешней среде, что дает повод для преобразования организации с помощью улучшения обслуживания клиентов.

## 1.7. Анализ внутренней среды

### Анализ микросреды

Микросреда – это элементы, с которыми непосредственно контактирует организация и она может оказывать свое влияние на эти элементы. Методикой выявляются пять сил, определяющих уровень конкуренции в отрасли и привлекательность отрасли в ведении бизнеса:

Вертикальные:

- Анализ рыночной власти потребителя;
- Анализ рыночной власти поставщиков;

Горизонтальные:

- Анализ уровня конкурентной борьбы;
- Анализ возможности появления новых конкурентов;
- Анализ угрозы появления новой продукции - заменителей;

**Вертикальные силы**

***1) Рыночная власть потребителей***

Потребителями продукции компании «Infinnity Solutions» являются федеральные организации, другие коммерческие компании или же физические лица, покупающие программные продукты компании через сервисы распространения мобильных приложений «Google Play» или «AppStore». Наиболее значимыми являются заказы от других компаний. Например, компания «Infinnity Solutions» разрабатывается средство связи между специалистами в области медицины с их клиентами. Так же важными для компании являются заказы от государства. Компанией были разработаны такие приложения как «Управдом ЖКХ» и «Хабинет» - сервис для родителей школьников. На данный момент, для компании не менее важны социально значимые сервисы, например «EcoRadar.online».

Компания «Infinnity Solutions» долгое время налаживала процессы внутри компании, выводила проекты на прибыльность

Конечными потребителями продукции фирмы «Infinnity Solutions» являются федеральные организации, различные сторонние компании, а также физические лица, осуществляющие покупку программного обеспечения через торговые площадки, такие как Google Play и App Store. Наиболее значимые заказы поступают фирме от государства: сотрудниками «Infinnity Solutions» был реализован ряд проектов для федеральных организаций: к примеру, «Управдом ЖКХ» и «Хабинет» - сервис для родителей школьников. Различные коммерческие организации, как правило, совершают разовые и менее крупные заказы.

У многих проектов компании есть общая целевая аудитория: приложение «Хабинет» и разрабатываемый сервис «Классная Москва» рассчитаны в основном на родителей школьников – высока вероятность, что этих пользователей привлекут оба программных продукта, так как они связаны с общей предметной областью «Школа». Следовательно, при разработке новых проектов или улучшении уже реализованных нужно учитывать все требования и пожелания потребителей, чтобы сохранить интерес у целевой аудитории к продукции фирмы.

Потеря основного потребителя может привести к существенным убыткам для компании. Необходимо предпринимать все необходимые меры для сохранения клиентской базы «Infinnity Solutions». Также важно привлекать новые целевые аудитории.

**Вывод:** у компании «Infinnity Solutions» существует множество крупных потребителей, заинтересованных продукцией фирмы и активно ее приобретающих. Потеря основного потребителя – федеральных организаций станет значительной для компании. Для устойчивого положения на рынке фирме необходимо не только сохранять, но и расширять клиентскую базу.

***2) Рыночная власть поставщиков***

Основным родом деятельности компании «Infinnity Solutions» является заказная разработка программной продукции. Для реализации проектов сотрудникам фирмы необходимо техническое задание от заказчика, компьютерное оборудование, программное обеспечение для разработки продукции и совместный интеллектуальный труд сотрудников. Так как в компании используется интеграционная платформа «InfinniPlatform» и инструмент для создания пользовательских интерфейсов «InfinniUI» собственной разработки, предприятие не нуждается в покупке дорогостоящих лицензий на подобное программное обеспечение.

Компания зависима от поставщиков компьютерного оборудования, программного обеспечения, необходимого для его корректной работы: операционная система, антивирусная система, серверное ПО и прочее. Также фирма может быть зависима от образовательных учреждений, поставляющих кадры компании: например, таким учреждением является Южно-Уральский государственный университет.

**Вывод:** компания «Infinnity Solutions» имеет достаточное количество поставщиков. Зависимость фирмы от одного поставщика незначительна.

**Характеристики внутренней среды**

Цель – выявление сильных и слабых сторон фирмы, оценка частей предприятия.

Для более конкретной оценки сильных и слабых сторон организации был выбран SNW-анализ. [2]

Расшифровка аббревиатуры:

S – сильная позиция;

N – нейтральная позиция;

W – слабая позиция.

Управленческие цели подхода:

1) сильные стороны как хороший ресурс организации сохранить + дополнительно усилить;

2) слабые стороны – т.е. плохой внутренний ресурс – устранить.

Основная причина добавления нейтральной стороны: для победы в конкурентной борьбе может оказаться достаточным состояние, когда данная конкретная организация относительно всех своих конкурентов по всем кроме одной ключевым позициям находится в состоянии N, и только по одному в состоянии S.

Данный анализ — это способ определить конкурентоспособность организации, при котором в роли нейтральной позиции лучше всего выбрать среднее рыночное состояние для определенной ситуации.

Оценка производится по результатам анализа внутренней среды.

Результаты анализа представлены на рисунке 5.

**1.8 Вывод:** У ООО «Infinnity Solutions» есть как сильные стороны (например, собственная среда разработки, качественное компьютерное оборудование, финансовая устойчивость, хороший имидж компании и высококвалифицированные специалисты), так и слабые – слабая система маркетинга. На основе данного анализа можно составить стратегию дальнейшего развития компании (усиление сильных сторон, ликвидация слабых).

**Экспертиза предприятия (методом SWOT-анализа)**

SWOT-матрица представлена в Таблице 3

| Факторы | S | W | N |
|---|---|---|---|
| **Производственный срез** | | | |
| Продукты | | | |
| Услуги | | | |
| Среда разработки | | | |
| **Управленческий срез** | | | |
| Тип структуры | | | |
| Вид управления | | | |
| **Финансовый срез** | | | |
| Финансовая устойчивость | | | |
| Заработные платы | | | |
| Квалификации работников | | | |
| Текучесть кадров | | | |
| **Информационный срез** | | | |
| Компьютерное оборудование | | | |
| Следование тенденциям | | | |
| Информационные потоки | | | |
| **Маркетинг** | | | |
| Имидж, репутация, качество | | | |
| Знание потребностей рынка | | | |
| Система маркетинга | | | |

Рисунок 5– результаты SWN анализа

В первом квадрате матрицы находятся ответы на вопрос: «Какие сильные стороны имеет предприятие?»

В нижнем левом квадрате приводится ответ на вопрос: «Какие слабые стороны есть в организации?»

В верхнем левом квадрате приведены возможности внешней среды, которые могут быть использованы предприятием.

Последний квадрат (нижний правый) содержит угрозы внешней среды, которые могут повлиять на организацию. [2]

Таблица 3-SWOT-матрица

| **S-Сильные стороны:** | **O-Возможности:** |
|---|---|
| 1.Собственная среда разработки<br>2.Качественное компьютерное оборудование<br>3.Хороший имидж компании<br>4.Высокая квалификация персонала<br>5.Финансовая устойчивость | 1.Рост спроса на технологии<br>2.Ввод санкций, действующий на IT-отрасль<br>3.Слабая зависимость от поставщиков<br>4.Мал шанс появления новых конкурентов |
| 1.Слабая система маркетинга | 1.Экономический кризис<br>2.Ужесточение требований разработки программного обеспечения<br>3.Ужесточение закона «О персональных данных»<br>4.Сильная зависимость от клиентов<br>5.Высокая конкуренция |
| **W-Слабые стороны:** | **T-Угрозы:** |

# ГЛАВА 2. РАЗРАБОТКА МОБИЛЬНОГО СЕРВИСА

## 2.1 Словарь терминов

Некоторые из упоминаемых терминов предметной области являются специализированными и требуют пояснений. Ниже дано описание основных понятий, используемых в описании предметной области.

**МП –** мобильное приложение сервиса Ecoradar.online

**Пользователь МП** – человек, загрузивший МП на своё устройство

**Сервер** – удалённая система, централизованно хранящая данные о загрязнении по городу Челябинску

### Определение типа информационной системы по МакФарлану

Роль информационных систем на предприятии определяется их функциональной направленностью и уровнем зрелости предприятия. [5] Это выражается во взаимосвязи стратегии и архитектуры ИТ, которую можно проследить, используя матрицу МакФарлана, изображенную на рисунке 8.

Анализ строится по двум оценкам:

1. Текущей зависимости предприятия от ИТ
2. Будущей зависимости предприятия от ИТ

***Зависимость будущей стратегии от применения ИТ***

| ***Текущая зависимость от применения ИТ*** | **Низкая** | **Высокая** |
|---|---|---|
| | **Поддержка (Вспомогательная ИС) 1** | **Постоянная готовность (Потенциальная ИС) 2** |

**Низкая**

| | |
|---|---|
| **Производство (Ключевая ИС) 3** | **Стратегия (Стратегическая ИС) 4** |

**Высокая**

Рисунок 8– Взаимосвязь стратегии и архитектуры ИТ

В качестве параметров для оценки (пятибалльной) текущей зависимости компании от ИТ могут быть рассмотрены следующие:

1.Зависимость повседневных операций от ИТ – 5

2.Надежность защиты информации – 5

3.Обеспечение обмена данными между ранее разработанными ИС и другими программными продуктами, функционирующими на предприятии – 4

4. Оперативность работы сотрудников – 5

5. Функциональная локализация информационной системы — 3

**Среднее значение – (5+5+4+5+3)/5 = 4,8 балла**

В качестве параметров для оценки (пятибалльной) будущей зависимости компании от ИТ могут быть рассмотрены следующие:

1.Возможность ИС стать конкурентным преимуществом – 5

2. Модификация существующего бизнеса за счет использования ИС – 4

3. Возможность консолидации информации – 4

4. Наличие специальных средств анализа состояния системы в процессе эксплуатации – 4

5. Усиление каналов сбыта за счет использования ИТ – 3

**Среднее значение – (5+4+4+4+3) /5= 4 балла**

**Вывод:** Если соотнести средние баллы, то можно сделать вывод, что информационная система в МРТ-клиники относится к классу «Стратегическая». Стратегические системы — те, от которых зависит бизнес, критичны для будущего делового успеха, конкуренции или стратегических преимуществ.

### 2.2 Исполнители и их задачи

Основные исполнители в проекте и их задачи представлены в таблице 5.

Таблица 5 – основные исполнители и их задачи

| Исполнитель | Описание | Задачи |
|---|---|---|
| Пользователь МП | Пользователь, заинтересованный в получении данных о загрязнении воздуха с помощью мобильного приложения Ecoradar.online | 1.Получение данных о загрязнении воздуха |

### 2.3 Выделение прецедентов

Прецедент – это набор сценариев использования, в котором каждый экземпляр сценария представляет собой последовательность действий, выполняемых системой для достижения ощутимого для конкретного исполнителя результата. [6]

Для наглядного представления прецедентов была составлена диаграмма прецедентов (Рисунок 9).

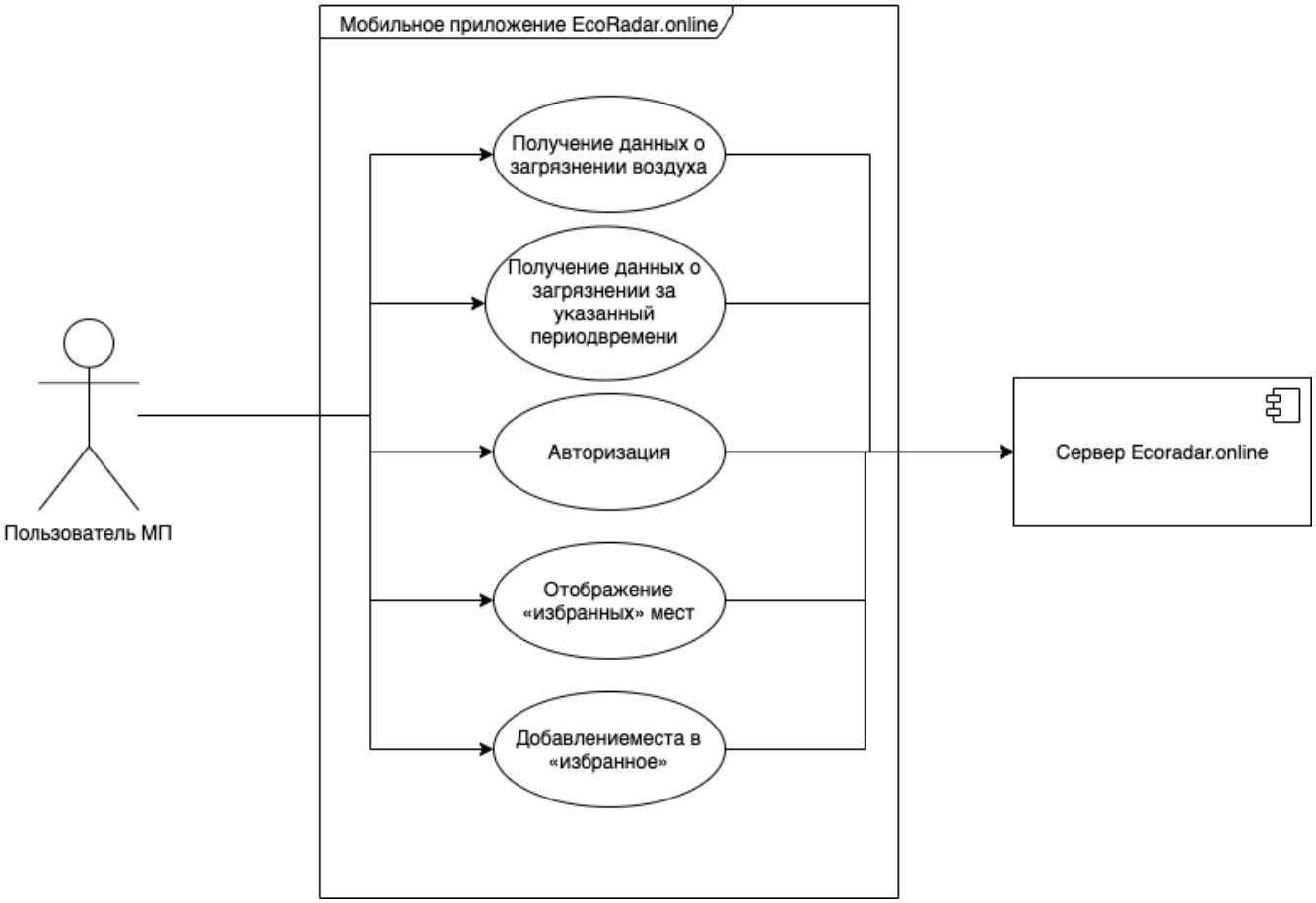


Рисунок 9 – Диаграмма прецедентов

**П1 Получение данных о загрязнении воздуха**

**Рамки:** Мобильное приложение EcoRadar.online

**Уровень:** Задача, определённая пользователем *(user-goal)*

**Основной исполнитель:** Пользователь МП

**Заинтересованные лица и их требования:**

- Пользователь МП. Быстрое получение актуальных данных по своей актуальной геопозиции, поскольку пользователь может перемещаться внутри города в течении дня.
- Компания. Минимизация затрат на каждую операцию пользователя такого типа т.к. они могут быть частыми и, если не будут оптимизированы, могут создать большую нагрузку на сервер. Получений актуальной статистики использования приложения для того, чтобы сконцентрироваться на

развитии и обеспечении качественной работы наиболее частых сценариев. Получение оперативной статистики по сбоям приложения для того, чтобы оперативно их исправлять.

**Предусловия:** Пользователь загрузил мобильное приложение

**Результаты (постусловия):** Пользователь МП получил актуальные данные о загрязнении вблизи своего местоположения. Статистика использования собрана. Сбои зарегестированы (если таковые были)

**Основной успешный сценарий:**

1. Пользователь открывает или разворачивает МП
2. МП запрашивает у пользователя разрешение на получение геолокационных данных о нём
3. МП получает данные о текущей геопозиции пользователя
4. МП делает запрос на актуальные данные с датчиков
5. Сервер обрабатывает запрос МП
6. Сервер получает актуальные данные о датчиках
7. Сервер отдаёт полученные данные МП
8. МП отображает полученные данные о загрязнении
9. МП акцентирует внимание пользователя на данных вблизи пользователя
10. МП отправляет в сервис сбора статистики данные о местоположении пользователя и о том, что он получил данные по загрязнению

**Расширения (альтернативные потоки):**

2а. Пользователь не дал разрешение получение геолокации, то при повторном заходе в МП:

1. МП повторно запросит данные о геолокации пользователя

3а. Геолокация пользователя недоступна или пользователь не дал разрешение на её получение:

1. После получения данных о загрязнении МП отображает их

2. Если ранее было получено местоположение пользователя, то происходит акцент внимания пользователя на данных вблизи него, иначе – акцент на данные в географическом центре города

4а. Приложение не может подключиться к серверу:

1. Если ранее были получены данные о загрязнении, МП отображает их. Если данные получены не были, то МП уведомляет пользователя об отсутствии сети

5а. Произошла ошибка со стороны сервера:

1. МП собирает данные об ошибке
2. МП собирает данные о своём текущем состоянии
3. МП сохраняет данные о своём состоянии локально
4. МП отправляет эти данные в сервис сбора статистики по сбоям

**Специальные требования:**

1. Данные о загрязнении должны отображаться в виде интерактивной карты
2. Приложение должно получать актуальные данные от сервера в течении 5 секунд при стабильном соединении с сетью скоростью не менее 1 мб/с

**П2 Получение данных о загрязнении за указанный период времени**

**Рамки:** Мобильное приложение EcoRadar.online

**Уровень:** Вспомогательная функция

**Основной исполнитель:** Пользователь МП

**Заинтересованные лица и их требования**

- Пользователь МП. Быстрое получение актуальных данных за указанный период времени, поскольку пользователь хочет увидеть динамику изменения уровня загрязнения в выбранном им месте.
- Компания. Минимизация затрат на каждую операцию пользователя такого типа т.к. если пользователь запросит данные такого рода один раз, то, скорее всего, он запросит данные за другие периоды в ближайшее время.

Если такие запросы не будут оптимизированы, они могут создать большую нагрузку на сервер.

**Предусловия:** Пользователь загрузил мобильное приложение и запросил актуальные данные по своей геопозиции

**Результаты (постусловия):** Пользователь МП получил актуальные данные о загрязнении за указанный период времени в указанной им геопозиции

**Основной успешный сценарий:**

1. Пользователь МП указывает временной промежуток, за который он хочет получить данные
2. Пользователь МП запрашивает данные по указанному временному промежутку
3. МП делает запрос на сервер на данные по указанному промежутку
4. Сервер получает данные по указанному промежутку
5. Сервер отдаёт эти данные
6. МП отображает пользователю данные, полученные от сервера
7. МП отправляет в сервис сбора статистики данные о запрошенном временном промежутке и о том, что пользователь запросил данные по нему

**Расширения (альтернативные потоки):**

2а. Сервер недоступен

1. МП показывает ошибку соединения

3а. Нет данных по текущему промежутку

1. Сервер отдаёт данные о том, что по переданному промежутку нет данных
2. МП отображает данные о том, что нет по указанному промежутку данных

**Специальные требования:**

1. Данные о загрязнении должны отображаться в виде интерактивной карты
2. Приложение должно получать актуальные данные от сервера в течении 5 секунд при стабильном соединении с сетью скоростью не менее 1 мб/с

**ПЗ Авторизация**

**Рамки:** Мобильное приложение EcoRadar.online

**Уровень:** Вспомогательная функция

**Основной исполнитель:** Пользователь МП

**Заинтересованные лица и их требования:**

- Пользователь МП. Синхронизация мест, добавленных в «избранное» между разными устройствами. Отсутствие доступа 3-х лиц к данным пользователя
- Компания. Безопасная передача данных пользователя от МП на сервер. Отсутствие доступа 3-х лиц к данным пользователя.

**Предусловия:** Пользователь загрузил мобильное приложение

**Результаты (постусловия):** Пользователь МП получил авторизован

**Основной успешный сценарий:**

1. Пользователь заходит в МП
2. Пользователь инициирует авторизацию
3. МП показывает пользователю интерфейс авторизации
4. Пользователь МП вводит свои идентификационные данные
5. МП передаёт идентификационные данные на сервер
6. Сервер валидирует идентификационные данные
7. Если валидация прошла успешно, сервер отдаёт МП данные о пользователе
8. МП показывает пользователю, что авторизация прошла успешно

**Расширения (альтернативные потоки):**

5а. Передача идентификационных данных не удалась

1. МП показывает пользователю ошибку авторизации

6а. Валидация данных не прошла успешно

1. Сервер отдаёт приложению о том, что идентификационные данные неверны
2. МП отображает пользователю информацию об этом

**Специальные требования:**

1. Сервер должен обрабатывать запрос на авторизацию в течении 10 секунд при стабильном соединении с сетью скоростью не менее 1 мб/с

**П4 Отображение «избранных» мест**

**Рамки:** Мобильное приложение EcoRadar.online

**Уровень:** Вспомогательная функция

**Основной исполнитель:** Пользователь МП

**Заинтересованные лица и их требования:**

- Пользователь МП. Отображение списка мест, добавленных им в «избранное». Синхронизация этого списка между всеми устройствами пользователя.
- Компания. Быстрая синхронизация точек, добавленных в «избранное» между всеми устройствами пользователя

**Предусловия:** Пользователь загрузил мобильное приложение, пользователь авторизован

**Результаты (постусловия):** Пользователь видит места, добавленные им в «избранное»

**Основной успешный сценарий:**

1. Пользователь МП инициирует переход к отображению избранных мест
2. МП запрашивает у сервера их список
3. Сервер получает данные об избранных местах по идентификационным данным пользователя МП
4. Сервер отдаёт эти данные МП
5. МП отображает их пользователю

**Расширения (альтернативные потоки):**

2а. Сервер недоступен

1. МП показывает пользователю ошибку соединения

**Специальные требования:**

1. Сервер должен обрабатывать запрос избранные места в течении 5 секунд при стабильном соединении с сетью скоростью не менее 1 мб/с

**П5 Добавление места в «избранное»**

**Рамки:** Мобильное приложение EcoRadar.online

**Уровень:** Вспомогательная функция

**Основной исполнитель:** Пользователь МП

**Заинтересованные лица и их требования:**

- Пользователь МП. Добавление выбранного места в список «избранных» мест
- Компания. Быстрая синхронизация точек, добавленных в «избранное» между всеми устройствами пользователя

**Предусловия:** Пользователь загрузил мобильное приложение, пользователь авторизован

**Результаты (постусловия):** Место, выбранное пользователем добавлен в список его «избранных мест»

**Основной успешный сценарий:**

1. Пользователь выбирает на интерактивной карте место, которое он хочет добавить в «избранное»
2. МП передаёт на сервер данные о выбранном пользователем месте
3. Сервер добавляет в список «избранных» мест пользователя указанное место
4. Сервер сообщает МП о том, что место успешно добавлено в «избранное»
5. МП отображает пользователю данные о том, что выбранное им место теперь находится в списке «избранных» им мест

**Расширения (альтернативные потоки):**

2а. Не удалось передать данные о выбранном пользователем месте на сервер

1. МП показывает пользователю ошибку соединения

**Специальные требования:**

1. Сервер должен обрабатывать запрос на добавление место в «избранное» в течении 5 секунд при стабильном соединении с сетью скоростью не менее 1 мб/с

**Модель предметной области**

Модель предметной области – это визуальное представление концептуальных классов или объектов реального мира в терминах предметной области. Такие модели называют также концептуальными моделями, моделями объектов предметной области, или объектными моделями анализа [6].

Для основного прецедента были определены классы и атрибуты для них (таблица 12). А также построена диаграмма классов (рисунок 16).


Рисунок 16 – Диаграмма классов

Таблица 6. Поля классов

| Класс | Имя поля | Тип поля | Обязательность |
|---|---|---|---|
| Пользователь | 1. id пользователя | 1. Целочисленный | 1. Да |
| Датчик | 1. id датчика | 1. Целочисленный | 1. Да |
| Показания | 1. Датчик<br>2. Уровень угрозы<br>3. Содержание P10<br>4. Норма P10<br>5. Содержание P2.5<br>6. Норма P2.5<br>7. Давление<br>8. Скорость ветра<br>9. Влажность | 1. Датчик<br>2. Уровень угрозы<br>3. Целочисленный<br>4. Целочисленный<br>5. Целочисленный<br>6. Целочисленный<br>7. Веществennй<br>8. Веществennй<br>9. Веществennй | 1. Да<br>2. Да<br>3. Да<br>4. Да<br>5. Да<br>6. Да<br>7. Да<br>8. Да<br>9. Да |

| | | | |
|---|---|---|---|
| | 10. Температура | 10. Веществennй | 10. Да |
| Местоположение | 1. Широта<br>2. Долгота | 1. Веществennй<br>2. Веществennй | 1. Да<br>2. Да |
| Уровень угрозы | 1. Высокий<br>2. Средний<br>3. Низкий | 1. Уровень угрозы<br>2. Уровень угрозы<br>3. Уровень угрозы | |
| Избранное | 1. Датчики | 1. Список датчиков | 1. Нет |
| Уведомления | 1. Заголовок<br>2. Текст | 1. Текст<br>2. Текст | 1. Да<br>2. Да |

### 2.4 Диаграмма деятельности

Диаграммы видов деятельности отображают последовательные и параллельные процессы. Они полезны для моделирования бизнес – процессов, последовательностей выполнения задач, потоков данных и сложных алгоритмов. [5]

Диаграмма деятельности, тесно связанная с прецедентами, поэтому для каждого рассмотрена отдельная диаграмма деятельности.

**Диаграмма деятельности получения списка показаний датчиков**

Представлена на рисунке 17.

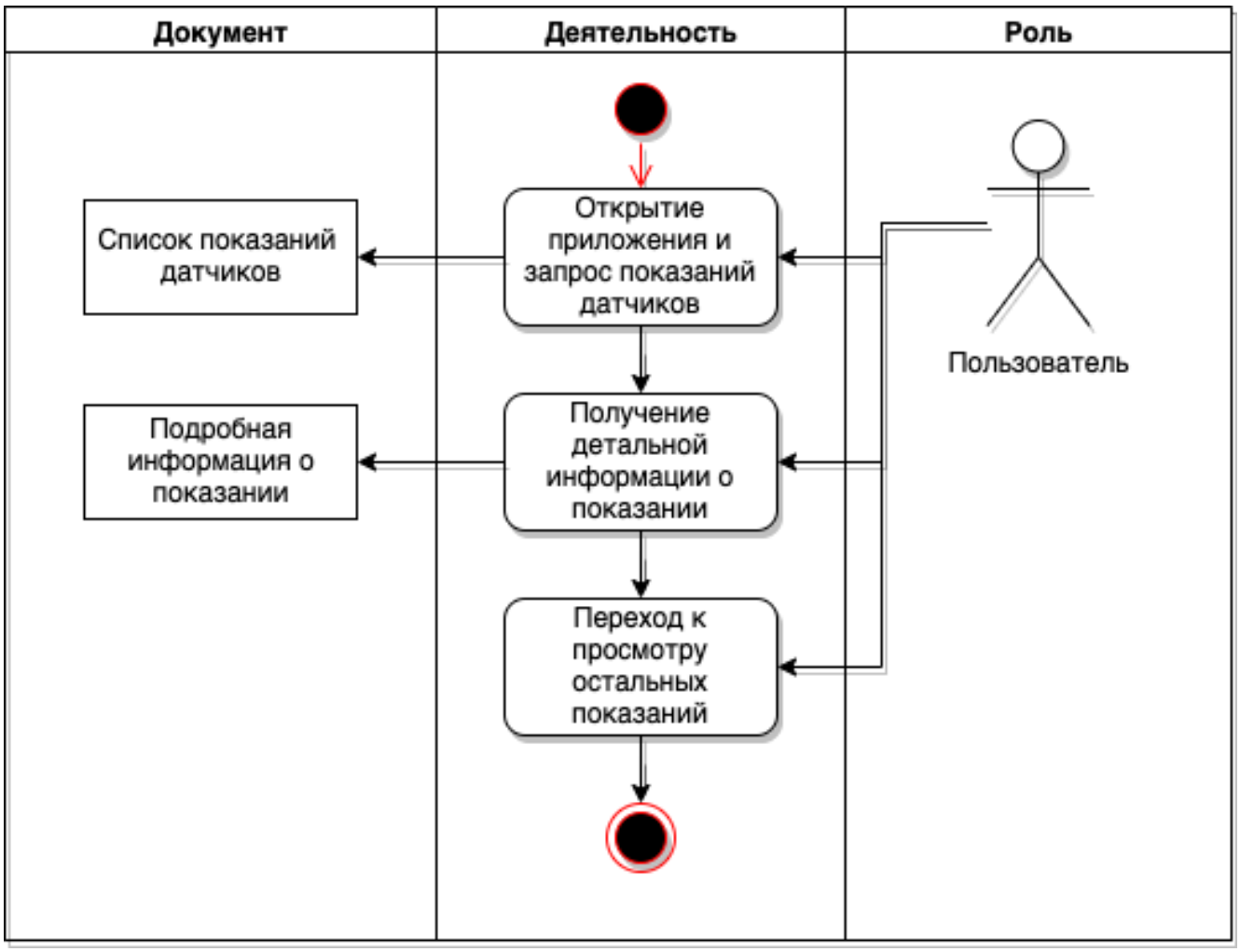


Рисунок 17 – Диаграмма деятельности получения списка показаний датчиков

**Диаграмма деятельности добавление датчика в «избранное»**

Представлена на рисунке 18.

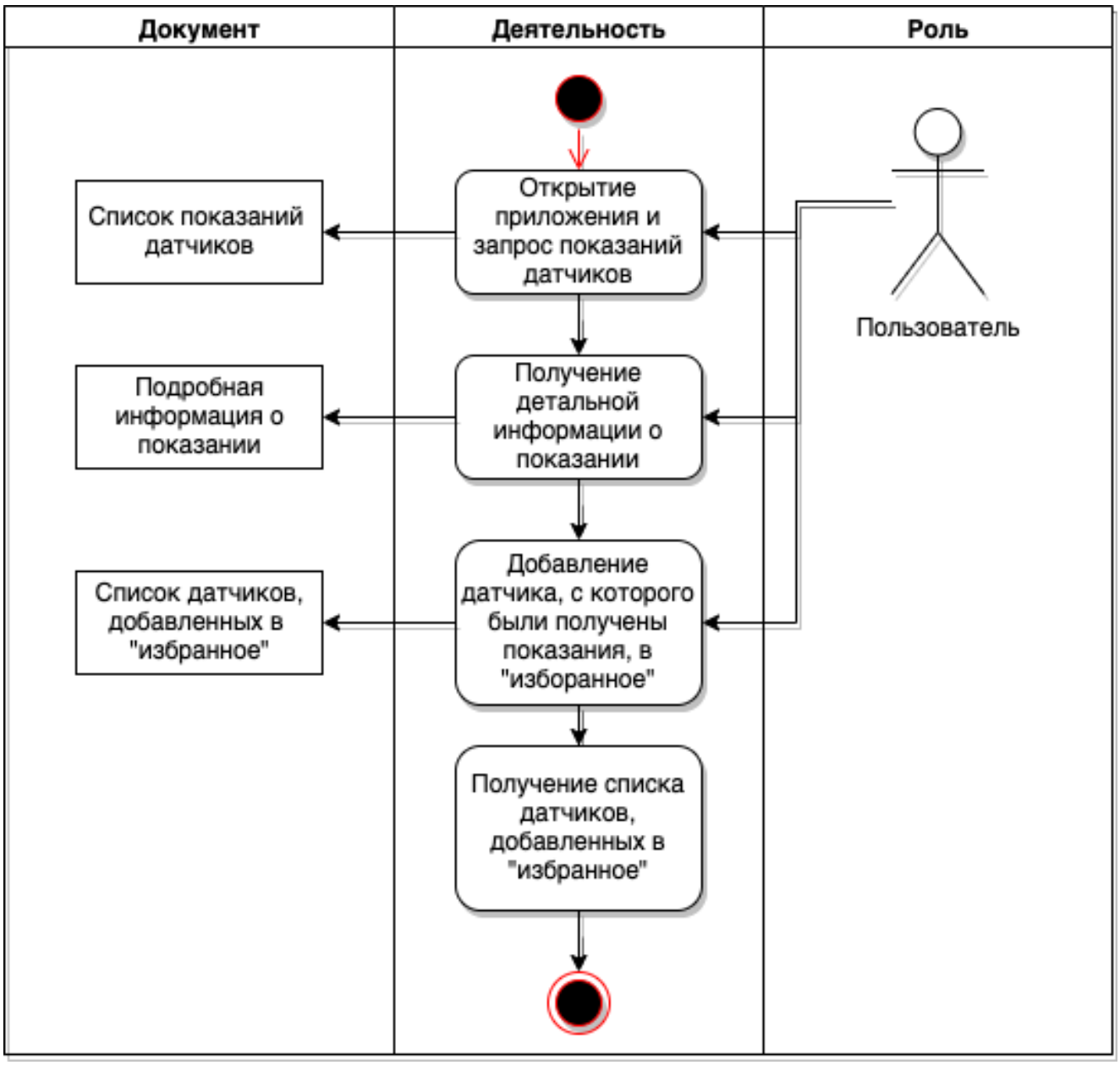


Рисунок 18 – Диаграмма деятельности добавление датчика в «избранное»

**Диаграмма деятельности удаление датчика из «избранного»**

Представлена на рисунке 19.

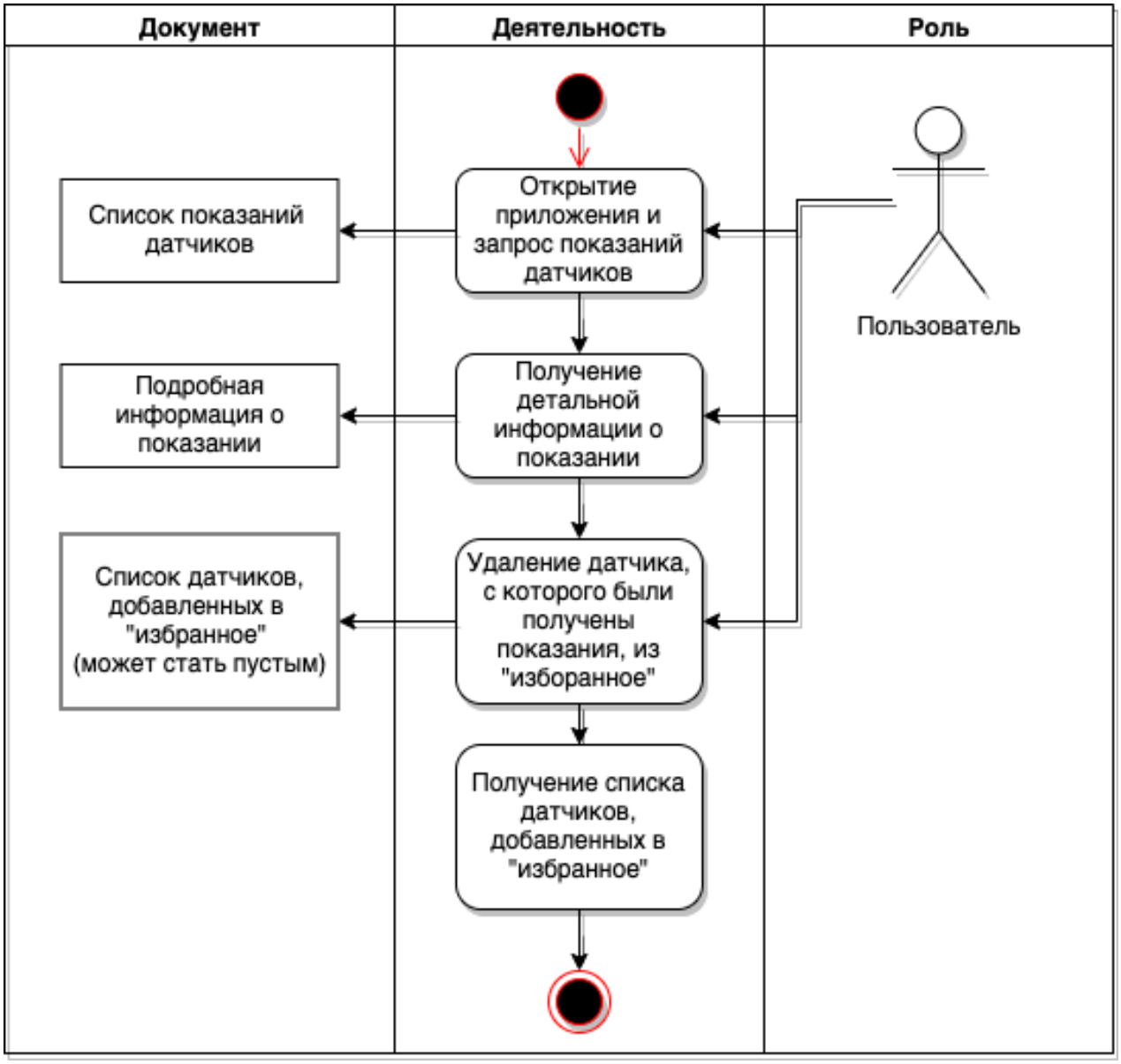


Рисунок 19 – Диаграмма деятельности удаление датчика из «избранного»

**Диаграмма деятельности получение показаний датчиков за указанный период времени**

Представлена на рисунке 20.

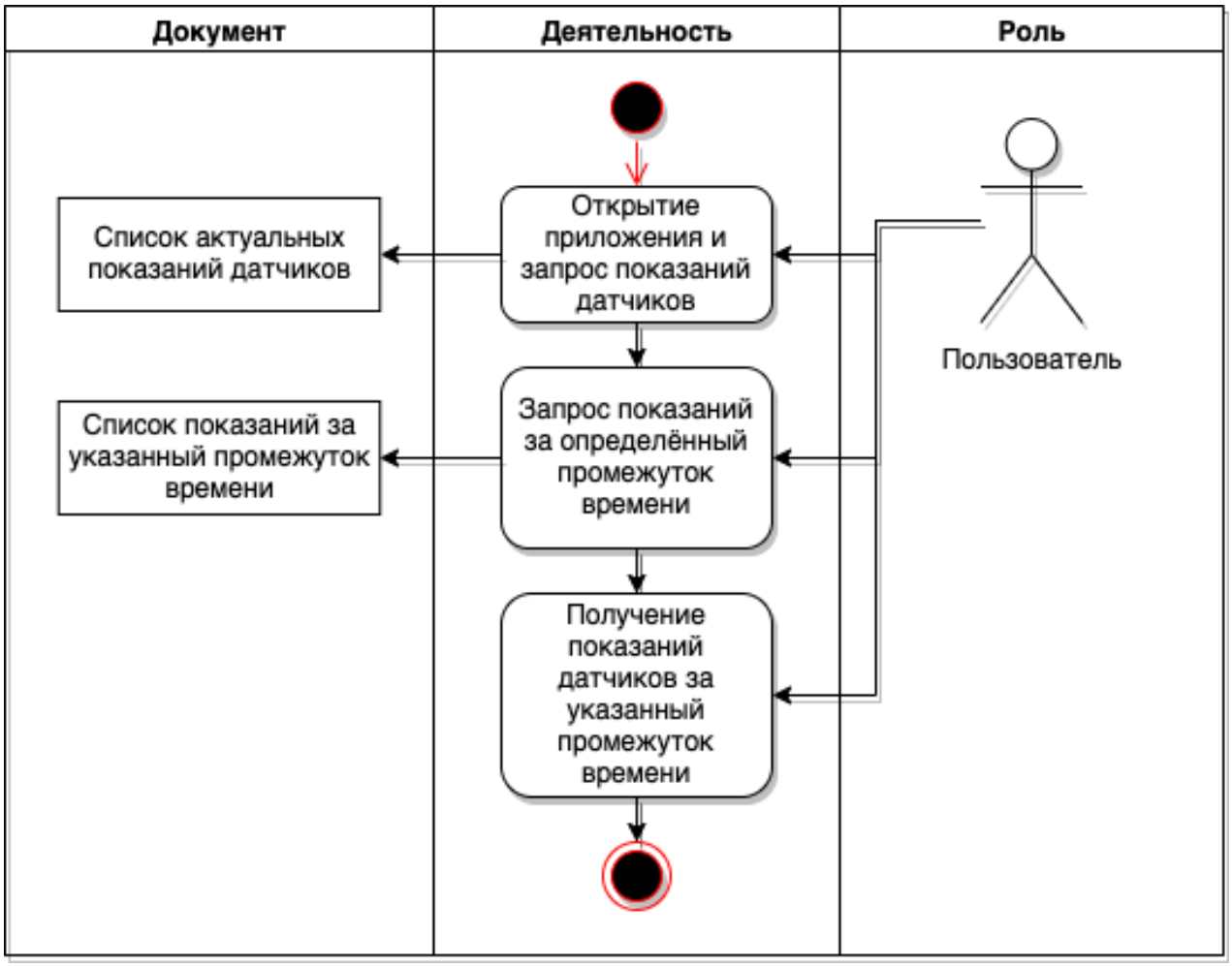


Рисунок 20 – Диаграмма деятельности получение показаний датчиков за указанный период времени

### 2.5 Серверная часть реализации проекта

Для реализации сервера был выбрал язык программирования Go. Это решение было основано на том, что Go:

1. Эффективен с точки зрения потребления ресурсов в сравнении с другими языками с автоматическим управлением памятью
2. Инструменты языка позволяют абстрагироваться от хранилища данных меньшими ограничениями на использование возможностей выбранной СУБД в сравнении с альтернативами

**Языки с автоматическим управлением памятью**

Есть множество преимуществ, которые языки с автоматическим управлением памятью предоставляют разработчикам программ. Они устраняют целые классы ошибок в программах, такие как попытка освободить память, на которую ссылается «висячий указатель», который всё ещё ссылается на уже освобождённую память, или даже хуже: ссылается на память, использующуюся в другом контексте. Они уменьшают шанс утечек памяти. Они упрощают создание структур данных, поддерживающих безопасный доступ из разных потоков. [7]

Рассмотрим поподробнее, как ручное управление памятью усложняет программный интерфейс модулей.

При модульной архитектуре, код программы декомпозируется в набор относительно независимых модулей. Модули могут принимать различные формы, такие как классы, подсистемы или сервисы [8].

Для управления зависимостями, мы можем разделить модуль на две части: интерфейс и реализацию. Интерфейс содержит всё, что необходимо знать для того, чтобы работать с данным модулем. Реализация содержит код, который удовлетворяет требованиям, описанным в интерфейсе [8].

Интерфейс модуля содержит 2 типа информации: формальную и неформальную. Формальная явно описана в коде. Её корректность может быть проверена автоматически [8].

Так же каждый интерфейс содержит и неформальную информацию, которая не может быть автоматически проверена. Более высокоуровневую информацию. К примеру, то, что функция «deleteFile» удаляет файл с именем, которое ей было передано в качестве одного из аргументов. Необходимый порядок вызова методов так же является частью неформальной части интерфейса [8]. Например, тот факт, что в языке Java для того, чтобы записать что-то в файл, его сначала необходимо создать с помощью функции «createNewFile». Такая информация может быть описана в коде в сигнатуре, а так же с помощью комментариев. При использовании большинства интерфейсов, неформальная информация намного важнее формальной [8].

Абстракция – это упрощённое представление сущности, которое опускает незначительные детали. Абстракция удобна, потому что упрощает управление сложными вещами [8].

При модульном программировании, абстракция модуля выражена в виде его интерфейса. Интерфейс представляет из себя его упрощённое представление. Детали реализации могут быть опущены при использовании модуля.

В определении абстракции слово «незначительные» является ключевым. Чем больше незначительных деталей исключено из абстракции, тем лучше. Однако деталь реализации может быть исключена из абстракции только в том случае, если она действительно незначительна. Есть 2 способа неправильно спроектировать абстракцию. Первый – описать в ней детали, которые, на самом деле, незначительны. Это увеличивает когнитивную нагрузку при использовании абстракции. Второй – исключить информацию, которая, на самом деле важна. Это приводит к тому, что у разработчика будет недостаточно информации для корректного использования модуля. Абстракция, которая опускает важные детали – ложная абстракция. Ключевым при разработке абстракции является понимание, что действительно важно для при использовании и минимизация количества информации, которая важна [8].

Рассмотрим с этой точки зрения функции, которые применяются для ручного управления памятью. Допустим, язык, который мы используем гарантирует, что переменная инициализирована при объявлении и использование объекта невозможно до выделения памяти под него. Предположим, функция «alloc» выделяет память под объект и возвращает его. Функция «dealloc» принимает в качестве аргумента объект и освобождает память, под него выделенную. Неформальный интерфейс этих функций содержит следующую информацию:

1. На возвращённом функцией «alloc» объекте должна быть вызвана функция «dealloc». Иначе будет утечка памяти
2. После вызова функции «dealloc» объект не должен использоваться
3. Функция «dealloc» не должна быть вызвана на уже освобождённой памяти, потому что она уже может использоваться другим объектом
4. Должно быть достаточное количество памяти для создания объекта

Теперь, предположим, что в нашем языке есть только функция «new», которая выделяет память под объект и возвращает его. Задачу освобождения памяти на себя берёт системы автоматического управления памятью. Неформальный интерфейс функции «new»:

1. Должно быть достаточное количество памяти для создания объекта

Как мы видим, добавление автоматического управления памятью упростило как формальный интерфейс (вместо двух функций теперь одна), так и неформальный. Благодаря использованию такой системы, упростится как разработка, так и дальнейшая поддержка кода программы.

**Потребление оперативной памяти объектами**

В качестве альтернатив Go, рассмотрим языки Java с виртуальной машиной HotSpot, Python, исполняемый с помощью PyPy и C# с CLR. Оптимизации компилятора учитывать не будем. На потребление памяти влияют различные

виды инлайнинга. Предположим, что они одинаково применимы ко всем перечисленным языкам.

Объект в оперативной памяти состоит из ссылки (или ссылок) на него, заголовочной информации, пустых промежутков в памяти, созданных при выравнивании и содержания объекта. Содержание аналогично для любых сред и языков. Предположим, что выравнивание так же одинаково для всех языков и платформ. Рассмотрим потребление памяти разными языками на 64-х битных операционных системах. Для Java включим сжатие указателей (UseCompressedOops).

Таблица 7. Размер объектов в оперативной памяти

| **Язык** | **Размер заголовков** | **Размер ссылок** | **Итого** |
|---|---|---|---|
| Go | 0 bytes [9] | 8 bytes | 8 bytes |
| Java | 16 bytes [10] | 4 bytes | 20 bytes |
| C# | 16 bytes [11] | 8 bytes | 24 bytes |
| Python | 48 bytes [12] | 8 bytes | 56 bytes |

Как итог, мы видим, что Go имеет наименьшее потребление памяти при использовании объектов по ссылке. Так же, стоит отметить, что в Go возможно использование экземпляров составных типов без ссылки, по значению. Это применимо, когда нет нужды передавать объект в несколько разных функций. Разница в размере заголовков обусловлена тем, что в объектах Python хранится информация о типе объекта. В языках со статической типизацией нет необходимости хранить такую информацию в каждом объекте. В Java и C# в объекте хранится информация о том, есть ли блокировка на данный объект, а также, прочая информация. К примеру, была ли применена оптимизация «сжатие строк» к объекту [14]. В Go внутри структуру хранятся только её поля. Если необходима блокировка, то для этого создаётся отдельный объект. При

реализации сервера для данного проекта, необходимости создать блокировку из кода приложения не разу не понадобилось.

Так же стоит учесть стартовое потребление памяти виртуальной машиной. Для этого запустим каждую виртуальную машину, подав ей на вход пустую программу.

Таблица 8. Начальное потребление памяти

| **Runtime** | **Начальное потребление памяти** |
|---|---|
| Go Runtime (Go) | ~ 7 mb |
| HotSpot (Java) | ~ 44 mb |
| CLR (C#) | ~ 40 mb |
| PyPy (Python) | ~ 84 mb |

Как мы видим, в данном тесте Go так же имеет минимальное потребление памяти среди аналогов. Это связано с тем, что runtime языка не содержит, к примеру, интерпретатора, необходимого для остальных языков для того, чтобы динамически компилировать и исполнять код.

Как итог, мы можем сказать, что Go является наиболее оптимальным выбором среди рассмотренных аналогов по потребления оперативной памяти.

**Потребления оперативной памяти на один поток выполнения**

Немалое влияние на общее потребление оперативной памяти, имеет количество памяти, выделяемое для одного потока выполнения. Под потоком выполнения будем иметь в виду наименьшую единицу обработки, изполнение которой может быть назначено используемой средой исполнения. Проведём рассчёты для Go, Java и C#. Несмотря на наличие потоков в Python, ручное управление или редко используется при разработке прикладных программ.

Таблица 9. Потребление памяти потоком управления

| **Единица обработки** | **Потребление памяти** |
|---|---|
| Goroutine (Go) | ~ 8kb |
| java.lang.Thread (Java) | ~ 1mb |
| System.Threading::Thread (C#) | ~ 4 mb |

Мы можем наблюдать, что Go является оптимальнее аналогов по потреблению памяти на один поток выполнения.

**Производительность**

Если рассмотреть результаты сравнения производительности в синтетических тестах Go и Java [15], Go и Python [16], Go и C# [17], то можно наблюдать, что в некоторых случаях самым быстрым является Go, в некоторых – C#. Из чего можно сделать вывод, что эти языки примерно равнозначны по производительности на стандартных синтетических тестах.

Наибольшее влияние на производительность web-приложений имеет не производительность вычисления языка, а скорость работы с базой данной и сетью [18]. Поэтому, чаще всего, можно пренебречь производительностью вычислений с помощью языка при разработке прикладных приложений.

Как итог сравнения эффективности использования ресурсов, можно сделать вывод, что Go является предпочтительным выбором по совокупности параметров.

**Разработка модулей управления СУБД**

Согласно правилу зависимостей (рисунок 21) из подхода «Чистая архитектура», модуль работы с СУБД является модулем низкого уровня [19]. Это означает, что изменения в нём, будут происходить, во-первых, чаще, во-вторых, причины этих изменений могут не зависеть от действий разработчиков прикладного программного обеспечения так как разработкой СУБД занимается другая команда разработки.

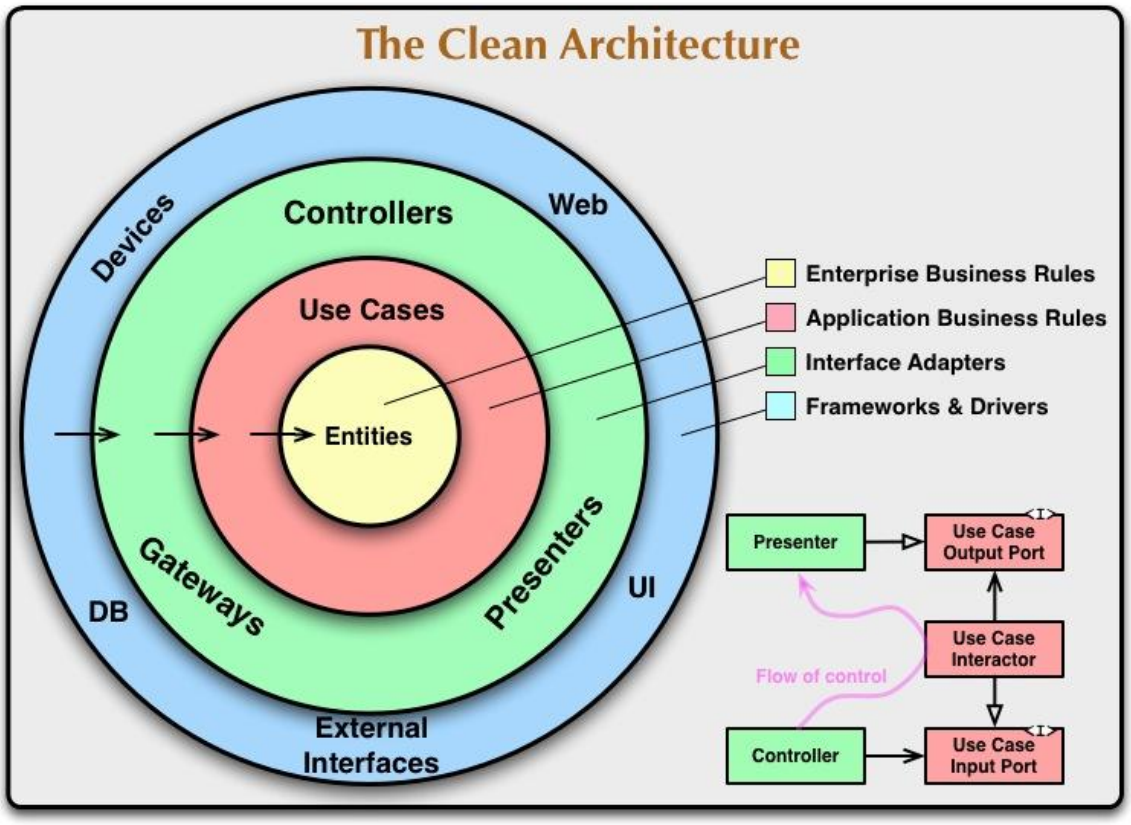


Рисунок 21, Правило зависимостей

Согласно *принципу инверсии зависимостей*, модули верхних уровней не должны зависеть от модулей нижних уровней. Оба типа модулей должны зависеть от абстракций [20].

Это означает, что код модуля бизнес-логики не должен напрямую ссылаться на код модуля работы с СУБД. В противном случае, обновление, смена или добавление нового источника данных может привести к изменению логики работы прикладной программы.

Согласно *принципу открытости/закрытости*, программные сущности должны быть открыты для расширения и закрыты для изменения [21].

В случае модуля работы с СУБД, это означает, что смена источника данных или добавление нового не должны привести к изменениям в коде модуля бизнес-логики.

Требования к технологии для работы с СУБД, сформулированные с учётом вышеперечисленных принципов:

1. Доступ к максимальному количеству возможностей СУБД
2. Возможность смены источника данных без внесения изменений в модуль бизнес-логики

В качестве СУБД была выбрана технология «PostgreSQL». Данный выбор обусловлен тем, что данная СУБД распространяется под открытой лицензией [22] и имеет более широкий список поддерживаемых стандартов [23], в сравнении с ближайшим аналогом «MySQL» [24]. Так же «PostgreSQL» входит в реестр отечественного программного обеспечения [25]. Это может сыграть роль в случае, если проект будет использоваться или дорабатываться государственными учреждениями.

Ниже приведён код создания схемы базы данных модуля добавления датчиков в «избранное» на языке PL/pgSQL.

```
CREATE TABLE users
(
    id          bigserial PRIMARY KEY
);

CREATE TABLE detectors
(
    id bigserial PRIMARY KEY
);

CREATE TABLE favorites
(
    id          bigserial PRIMARY KEY,
    detector_id bigint NOT NULL REFERENCES detectors
        ON UPDATE CASCADE ON DELETE CASCADE,
    user_id     bigint NOT NULL REFERENCES users
        ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE UNIQUE INDEX ON favorites (detector_id, user_id);
```

Рассмотрим код на языке Go, позволяющий изменить список датчиков, находящихся в «избранном». В модуле бизнес-логики находится следующий код:

```
type Favorites struct {
   DetectorIds []int64
}

func NewFavorites(
    detectorIds []int64,
) *Favorites {
    return &Favorites{
        DetectorIds: detectorIds,
    }
}

type FavoritesRepo interface {
   Update(userId int64, detectorIds []int64) (*client.Favorites, error)
}

favorites, err := repo.Update(userId, detectorIds)
if err != nil {
    // handle err
}
```

В модуле управления СУБД находится следующий код:

```
type FavoritesRepo struct {
   *dbms.DB
}

func (d *FavoritesRepo) Update(
   userId int64, detectorIds []int64
) (*client.Favorites, error) {

   tx, err := d.Begin()
   if err != nil {
      return nil, err
   }

   _, err = tx.Exec(`
DELETE FROM favorites
WHERE user_id = $1`,
      userId,
   )
   if err != nil {
      return nil, err
   }
```

```
    for _, detectorId := range detectorIds {
        _, err := d.Exec(`
INSERT INTO favorites (detector_id, user_id)
VALUES ($1, $2)`,
            detectorId, userId,
        )
        if err != nil {
            return nil, err
        }
    }

    err = tx.Commit()
    if err != nil {
        return nil, err
    }

    return getFavorites(userId, d.DB)
}

func getFavorites(
    userId int64, d *dbms.DB,
) (*client.Favorites, error) {
    favorites := &client.Favorites{}
    err := d.QueryRow(`
SELECT ARRAY(
     SELECT detector_id
     FROM favorites
     WHERE user_id = $1)`,
        userId,
    ).Scan(
        d.Array(&favorites.DetectorIds),
    )
    if err != nil {
        return nil, err
    }
    return favorites, nil
}
```

В качестве языка управления СУБД в данном коде используется язык pgSQL. Использование данного языка позволяет получить доступ к максимальному количеству возможностей «PostgreSQL» [26]. Так же, в интерфейсе драйвера для работы с СУБД, поддерживающими SQL из стандартной

библиотеки Go присутствует метод «Scan». Использование данного метода позволяет поместить значения, полученные из базы данных сразу в объект, не создавая дополнительные локальные переменные [27]. Интерфейсы драйверов из стандартных библиотек языков C#, Java и Python, не предоставляют такой возможности [28, 29, 30].

Альтернативой работы с интерфейсом драйвера базы данных напрямую являются ORM. ORM (Object-Relational Mapping) – это технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Такие технологии позволяют получить результаты запросов сразу в виде объектов, но затрудняют использование plSQL в качестве языка запросов к СУБД [31, 32, 33].

Для того чтобы проверить соответствие реализации второму требования, было реализовано сохранение данных в альтернативный источник данных – файловую систему в формате «JSON».

```
type FavoritesRepo struct {
}

func (d *FavoritesRepo) Update(
   userId int64, detectorIds []int64,
) (*client.Favorites, error) {

   detectorIdsJson, err := json.Marshal(detectorIds)
   if err != nil {
      return nil, err
   }

   err = ioutil.WriteFile(
      fmt.Sprintf("%d.favorites.json", userId),
      detectorIdsJson,
      os.ModeAppend,
   )
   if err != nil {
      return nil, err
   }
```

```
    return client.NewFavorites(detectorIds), nil
}
```

Возможность реализовать новый источник данных без изменения кода модуля бизнес-логики означает, что второе требование к технологии управления источниками данных так же удовлетвворено.

Таким образом, при использовании языка Go были удовлетворены все представленне требования к модулю управления СУБД.

### 2.5 Клиентская часть реализации проекта

Для технической реализации клиента были выбраны язык Dart и технология «Flutter». Выбор обоснован тем, что данные технологии позволяют сделать общими около 95% кодовой базы [34] для приложений под ОС Android и ОС iOS.

Flutter — SDK с открытым исходным кодом для создания мобильных приложений от компании Google. Он используется для разработки приложений под Android и iOS, а также это пока единственный способ разработки приложений под Google Fuchsia.

Движок написан преимущественно на C++, он поддерживает низкоуровневый рендеринг с помощью графической библиотеки Google Skia. А также имеет возможность взаимодействовать с платформозависимыми SDK под Android и iOS.

Дизайн пользовательского интерфейса приложений Flutter обычно включает в себя использование и/или создание различных виджетов. Виджет Flutter представляет собой неизменяемое описание какой-либо части пользовательского интерфейса. Все графические объекты, включая текст, формы и анимацию, создаются с помощью виджетов. Комбинированием простых виджетов создаются комплексные виджеты.

#### Система версионирования Dart

Язык Dart поддерживает современную систему версионирования стороних пакетов. Она состоит из двух частей: формул версий, описанных в файле «pubspec.yaml» и зафиксированных версий, описанных в файле «pubspec.lock».

Блок описания версии приложения:

```
version: 1.0.0+1
```

Данный блок указывает, какая версия приложения будет отображаться пользователю.

Блок описания версии стандартного пакета разработки:

```
environment:
  sdk: ">=2.3.0 <3.0.0"
```

Описание версии содержит минимальную возможную версию языка Dart, которую может использовать приложение.

SDK (от англ. software development kit) — набор средств разработки, который позволяет специалистам по программному обеспечению создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы, компьютерной системы, игровых консолей, операционных систем и прочих платформ.

Программист, как правило, получает SDK непосредственно от разработчика целевой технологии или системы. Часто SDK распространяется через Интернет. Многие SDK распространяются бесплатно для того, чтобы побудить разработчиков использовать данную технологию или платформу.

Поставщики SDK иногда подменяют слово «software» в словосочетании «software development kit» на более точное слово. Например, Microsoft и Apple предоставляют Driver Development Kit (DDK) для разработки драйверов

устройств, PalmSource называет свой инструментарий для разработки PalmOS Development Kit (PDK), a Oracle — Java Development Kit (JDK).

Ниже описан блок версий сторонних зависимостей.

```
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^0.1.2
  google_maps_flutter: ^0.5.21+7
  google_sign_in: ^4.1.1
  http: ^0.12.0+1
```

В блоке сторонних зависимостей описан используемый в приложении набор библиотек.

Версии сторонних зависимостей описаны в виде формул. Формулы позволяют описать несколько версий вместо какой-то одной конкретной.

К примеру, символ «^» обозначает, что минорная версия может быть автоматически заменена на более новую.

Для версионирования пакетов Dart используется система «Semver» или семантическое версионирование.

Правила семантического версионирования описаны ниже:

1. ПО, использующее Семантическое Версионирование, должно объявить публичный API. Этот API может быть объявлен самим кодом или существовать строго в документации. Как бы ни было это сделано, он должен быть точным и исчерпывающим.
2. Обычный номер версии должен иметь формат X.Y.Z, где X, Y и Z — неотрицательные целые числа и не должны начинаться с нуля. X — мажорная версия, Y — минорная версия и Z — патч-версия. Каждый элемент должен увеличиваться численно. Например: 1.9.0 ->1.10.0 -> 1.11.0.

3. После релиза новой версии пакета содержание этой версии не должно быть модифицировано. Любые изменения должны быть выпущены как новая версия.
4. Мажорная версия ноль (0.y.z) предназначена для начальной разработки. Всё может измениться в любой момент. Публичный API не должен рассматриваться как стабильный.
5. Версия 1.0.0 определяет публичный API. После этого релиза номера версий увеличиваются в зависимости от того, как изменяется публичный API.
6. Патч-версия Z (x.y.Z | x > 0) должна быть увеличена только если содержит обратно совместимые баг-фиксы. Определение баг-фикс означает внутренние изменения, которые исправляют некорректное поведение.
7. Минорная версия (x.Y.z | x > 0) должна быть увеличена, если в публичном API представлена новая обратно совместимая функциональность. Версия должна быть увеличена, если какая-либо функциональность публичного API помечена как устаревшая (deprecated). Версия может быть увеличена в случае реализации новой функциональности или существенного усовершенствования в приватном коде. Версия может включать в себя изменения, характерные для патчей. Патч-версия должна быть обнулена, когда увеличивается минорная версия.
8. Мажорная версия X (X.y.z | X > 0) должна быть увеличена, если в публичном API представлены какие-либо обратно несовместимые изменения. Она может включать в себя изменения, характерные для уровня минорных версий и патчей. Когда увеличивается мажорная версия, минорная и патч-версия должны быть обнулены.
9. Предрелизная версия может быть обозначена добавлением дефиса и серией разделённых точкой идентификаторов, следующих сразу за патч-версией. Идентификаторы ДОЛЖНЫ содержать только ASCII буквенно-цифровые символы и дефис [0-9A-Za-z-]. Идентификаторы не должны быть пустыми. Числовые идентификаторы не должны начинаться с нуля. Предрелизные

версии имеют более низкий приоритет, чем соответствующая релизная версия. Предрелизная версия указывает на то, что эта версия не стабильна и может не удовлетворять требованиям совместимости, обозначенными соответствующей нормальной версией. Примеры: 1.0.0-alpha, 1.0.0-alpha.1, 1.0.0-0.3.7, 1.0.0-x.7.z.92.

10. Сборочные метаданные могут быть обозначены добавлением знака плюс и ряда разделённых точкой идентификаторов, следующих сразу за патчем или предрелизной версией. Идентификаторы должны содержать только ASCII буквенно-цифровые символы и дефис [0-9A-Za-z-]. Идентификаторы не должны быть пустыми. Сборочные метаданные следует игнорировать, когда определяется старшинство версий. Поэтому два пакета с одинаковой версией, но разными сборочными метаданными, рассматриваются как одна и та же версия. Примеры: 1.0.0-alpha+001, 1.0.0+20130313144700, 1.0.0-beta+exp.sha.5114f85.
11. Приоритет определяет, как версии соотносятся друг с другом, когда упорядочиваются. Приоритет версий должен рассчитываться путём разделения номеров версий на мажорную, минорную, патч и предрелизные идентификаторы. Именно в такой последовательности (сборочные метаданные не фигурируют в расчёте). Приоритет определяется по первому отличию при сравнении каждого из этих идентификаторов слева направо: Мажорная, минорная и патч-версия всегда сравниваются численно. Пример: 1.0.0 < 2.0.0 < 2.1.0 < 2.1.1. Когда мажорная, минорная и патч-версия равны, предрелизная версия имеет более низкий приоритет, чем нормальная версия. Пример: 1.0.0-alpha < 1.0.0. Приоритет двух предрелизных версий с одинаковыми мажорной, минорной и патч-версией должны быть определены сравнением каждого разделённого точкой идентификатора слева направо до тех пор, пока различие не будет найдено следующим образом: идентификаторы, состоящие только из цифр, сравниваются численно; буквенные идентификаторы или дефисы сравниваются лексически в ASCII-порядке. Численные идентификаторы всегда

имеют низший приоритет, чем символьные. Больший набор предрелизных символов имеет больший приоритет, чем меньший набор, если сравниваемые идентификаторы равны. Пример: 1.0.0-alpha < 1.0.0-alpha.1 < 1.0.0-alpha.beta < 1.0.0-beta < 1.0.0-beta.2 < 1.0.0-beta.11 < 1.0.0-rc.1 < 1.0.0.

Далее описана ссылка на сгенерированный кодогенератором клиент на языке Dart. Версия клиента выводится автоматически.

```
client:
  git: https://gitlab.com/ecoradar/dart_api_client.git
```

Данный клиент сгенерирован на основе спецификации API.

Помимо описания версий в виде формул, существует так же описание версий конкретных версий зависимостей. Он описан в файле «pubspec.lock»

Ниже приведены зависимости для исполняемой среды языка Dart:

```
archive:
  dependency: transitive
  description:
    name: archive
    url: "https://pub.dartlang.org"
  source: hosted
  version: "2.0.11"
```

Ниже приведены зависимости с конкретными версиями для асинхронной инфраструктуры:

```
args:
  dependency: transitive
  description:
    name: args
    url: "https://pub.dartlang.org"
  source: hosted
  version: "1.5.2"
async:
  dependency: transitive
  description:
    name: async
```

```
    url: "https://pub.dartlang.org"
  source: hosted
  version: "2.4.0"
boolean_selector:
  dependency: transitive
  description:
    name: boolean_selector
    url: "https://pub.dartlang.org"
  source: hosted
  version: "1.0.5"
charcode:
  dependency: transitive
  description:
    name: charcode
    url: "https://pub.dartlang.org"
  source: hosted
  version: "1.1.2"
```

Ниже приведены зависимости с конкретными версиями для карт Google:

```
google_maps_flutter:
  dependency: "direct main"
  description:
    name: google_maps_flutter
    url: "https://pub.dartlang.org"
  source: hosted
  version: "0.5.21+15"
```

Ниже приведены зависимости с конкретными версиями для авторизации с помощью Google:

```
google_sign_in:
  dependency: "direct main"
  description:
    name: google_sign_in
    url: "https://pub.dartlang.org"
  source: hosted
  version: "4.1.1"
google_sign_in_platform_interface:
  dependency: transitive
  description:
    name: google_sign_in_platform_interface
    url: "https://pub.dartlang.org"
  source: hosted
  version: "1.0.3"
```

```
google_sign_in_web:
  dependency: transitive
  description:
    name: google_sign_in_web
    url: "https://pub.dartlang.org"
  source: hosted
  version: "0.8.3"
```

Ниже приведены зависимости с конкретными версиями для работы с сетью:

```
http:
  dependency: "direct main"
  description:
    name: http
    url: "https://pub.dartlang.org"
  source: hosted
  version: "0.12.0+4"
http_parser:
  dependency: transitive
  description:
    name: http_parser
    url: "https://pub.dartlang.org"
  source: hosted
  version: "3.1.3"
image:
  dependency: transitive
  description:
    name: image
    url: "https://pub.dartlang.org"
  source: hosted
  version: "2.1.4"
```

Ниже приведены зависимости с конкретными версиями для сгенерированного на основе API клиента для Dart:

```
client:
  dependency: "direct main"
  description:
    path: "."
    ref: HEAD
    resolved-ref: "5226dab705f8f97c61f94c5052db808e270b0757"
    url: "https://gitlab.com/ecoradar/dart_api_client.git"
  source: git
  version: "0.31.0"
```

Благодаря современной системе версионирования сторонних зависимостей, поддержка актуальности сторонних зависимостей для приложения, написанного на языке Dart становится простой.

**Декларативный подход к описанию пользовательского интерфейса**

Для построения UI во Flutter используется декларативный подход на основе виджетов. Для еще большего прироста в скорости работы интерфейса виджеты перерисовываются по необходимости — только когда в них что-то изменилось

Ниже приведён пример виджета:

```
class IndicationsMap extends StatefulWidget {
  @override
  State<IndicationsMap> createState() => IndicationsMapState();
}

class IndicationsMapState extends State<IndicationsMap> {
  GoogleSignIn _googleSignIn = GoogleSignIn(
    scopes: [
      'email',
      'https://www.googleapis.com/auth/contacts.readonly',
    ],
  );

  static final CameraPosition _defaultPosition = CameraPosition(
    target: LatLng(55.106112, 61.485833),
    zoom: 14.4746,
  );

  final _api = DefaultApi(ApiClient(
    basePath: "http://138.197.196.162:8484/api/client/v1",
  ));

  final _favoritesHolder = ValueHolder<Favorites>();
  Favorites _favorites;

  var _indications = List<Indication>();

  final _userHolder = ValueHolder<User>();
  User _user;
```

```
@override
void initState() {
  _favoritesHolder.subscribe((favorites) {
    setState(() {
      _favorites = favorites;
    });
  });

  _userHolder.subscribe((user) {
    setState(() {
      _user = user;
    });
    _api.getFavorites(userId: user.id).then((favorites) {
      _favoritesHolder.put(favorites);
    }).catchError((error) {
      debugPrint(error);
    });
  });

  _fetchIndications();

  Stream.periodic(Duration(minutes: 3)).listen((_) {
    _fetchIndications();
  });
  super.initState();
}

void _fetchIndications() {
  _api.getRecentIndications().then((indications) {
    setState(() {
      _indications = indications;
    });
  }).catchError((error) {
    debugPrint(error);
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(
      children: <Widget>[
        if (_indications.isNotEmpty) ...[
          GoogleMap(
            mapType: MapType.normal,
```

```
            initialCameraPosition: _defaultPosition,
            markers: _indications
                .map((indication) => _makeMarker(indication))
                .toSet(),
            circles: _indications
                .map((indication) => _makeCircle(indication))
                .toSet(),
          ),
          if (_user == null)
            Container(
              child: FloatingActionButton.extended(
                label: Text("В о й т и"),
                onPressed: () {
                  _googleSignIn.signIn().then((resp) {
                    _api.auth(AuthRequest()..id = resp.id).then((user) {
                      _userHolder.put(user);
                    });
                  }).catchError((error) {
                    debugPrint(error);
                  });
                },
              ),
              alignment: Alignment.topRight,
              padding: EdgeInsets.only(right: 20, top: 40),
            )
        ],
        if (_indications.isEmpty)
          Center(
            child: CircularProgressIndicator(
              backgroundColor: Colors.cyanAccent,
            ),
          )
      ],
    ),
  );
}

Marker _makeMarker(Indication indication) {
  final isInFavorites = _getIsInFavorites(_favorites, indication);
  BitmapDescriptor icon = BitmapDescriptor.defaultMarkerWithHue(
    isInFavorites ? BitmapDescriptor.hueRed : BitmapDescriptor.hueCyan,
  );
  return Marker(
    markerId: MarkerId("${indication.detector.id}"),
    position: LatLng(
```

```
        indication.detector.location.lat,
        indication.detector.location.lon,
      ),
      icon: icon,
      onTap: () {
        _showModalSheet(indication);
      },
    );
  }

  void _showModalSheet(Indication indication) {
    showModalBottomSheet(
        context: context,
        builder: (builder) {
          return DetailsSheet(indication, _favoritesHolder, _userHolder, () {
            final request =
                _switchedFavoritesRequest(_favorites, indication, _user);
            _api.updateFavorites(request).then((favorites) {
              _favoritesHolder.put(favorites);
            });
          });
        });
  }

  Circle _makeCircle(Indication indication) {
    Color color;
    switch (indication.dangerLevel) {
      case DangerLevel.low:
        color = Colors.lightGreen;
        break;
      case DangerLevel.medium:
        color = Colors.orangeAccent;
        break;
      case DangerLevel.high:
        color = Colors.redAccent;
        break;
    }
    return Circle(
      circleId: CircleId("${indication.detector.id}"),
      center: LatLng(
        indication.detector.location.lat,
        indication.detector.location.lon,
      ),
      fillColor: color.withAlpha(155),
      radius: 1200.0,
```

```
      strokeWidth: 0,
    );
  }
}
```

Вместо того, чтобы менять состояние существующих виджетов, подход, используемый при разработке с использованием Flutter, позволяет отрисовать изменившийся виджет заново.

Благодаря используемому подходу, время разработки при помощи Flutter сокращается, а так же уменьшается общая сложность программы так как каждое возможное состояние пользовательского интерфейса явно описано в коде.

### 2.6 Архитектура приложения

При проектировании архитектуры, был использован подход «Чистая Архитектура». Чистая архитектура основана на соблюдении пяти принципом проектирования.

**Принцип единственной ответственности** — принцип ООП, обозначающий, что каждый объект должен иметь одну ответственность и эта ответственность должна быть полностью инкапсулирована в класс. Все его поведения должны быть направлены исключительно на обеспечение этой ответственности.

**Принцип открытости/закрытости** означает, что программные сущности должны быть:

1. Открыты для расширения: означает, что поведение сущности может быть расширено путём создания новых типов сущностей.
2. Закрыты для изменения: в результате расширения поведения сущности, не должны вноситься изменения в код, который эта сущность использует.

Это особенно значимо в производственной среде, когда изменения в исходном коде потребуют проведение пересмотра кода, модульного тестирования и других подобных процедур, чтобы получить право на использование его в программном

продукте. Код, подчиняющийся данному принципу, не изменяется при расширении и поэтому не требует таких трудозатрат.

**Принцип разделения интерфейса.** Принцип разделения интерфейсов говорит о том, что слишком «толстые» интерфейсы необходимо разделять на более маленькие и специфические, чтобы программные сущности маленьких интерфейсов знали только о методах, которые необходимы им в работе. В итоге, при изменении метода интерфейса не должны меняться программные сущности, которые этот метод не используют.

**Принцип подстановки Барбары Лисков.** Функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом.

**Принцип инверсии зависимостей** Модули верхних уровней не должны зависеть от модулей нижних уровней. Оба типа модулей должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

Принцип инверсии зависимостей подразумевает разделение сущностей на сущности более низкого и сущности более высокого уровня. На рисунке 21 изображена схема правила зависимостей, показывающая, каким образом должно быть произведено разделение сущностей на низкоуровневые и высокоуровневые.


Рисунок 22. Правило зависимости

Согласно правилу зависимостей, приложение было разделено на модули следующим образом (Рисунок 23):

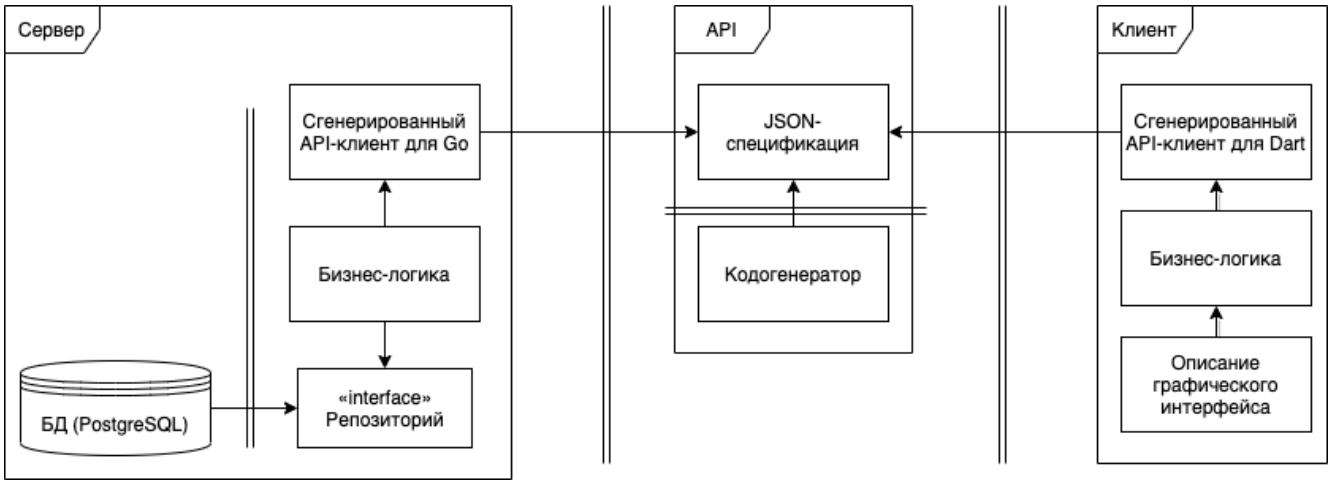


Рисунок 23. Архитектура приложения Ecoradar.online

Разработка архитектуры программного обеспечения — это искусство проведения разделяющих линий, которые я называю границами. Границы отделяют программные — даже до появления первого программного кода. Другие проводятся намного позже. Границы, проводимые на ранних этапах, призваны отложить принятие решений на как можно долгий срок и предотвратить загрязнение основной бизнес-логики этими решениями. Напомню, что целью архитектора является минимизация трудозатрат на создание и сопровождение системы. Что может помешать достижению этой цели? Зависимость — и особенно зависимость от преждевременных решений. Какие решения можно назвать преждевременными? Решения, не имеющие ничего общего с бизнес-требованиями — вариантами использования — системы. К ним можно отнести решения о выборе фреймворка, базы данных, веб-сервера, вспомогательных библиотек, механизма внедрения зависимостей и т.п. В хорошей архитектуре подобные решения носят вспомогательный характер и откладываются на потом.

Хорошая архитектура не зависит от таких решений. Хорошая архитектура позволяет принимать эти решения в самый последний момент без существенного влияния на саму архитектуру [35].

На рисунке 23 архитектурные границы обозначены двойными чертами.

«Отделять линиями нужно все, что не имеет значения. Графический интерфейс не имеет значения для бизнес-правил, поэтому между ними нужно провести границу. База данных не имеет значения для графического интерфейса, поэтому между ними нужно провести границу. База данных не имеет значения для бизнес-правил, поэтому между ними нужно провести границу» [36].

Так как база данных и пользовательский интерфейс являются модулями низкого уровня, архитектурная граница между ними и бизнес-логикой пересекается только в сторону бизнес-логики.

Ниже приведён исходный код модуля работы бизнес-логики для получения показаний с датчиков:

```
type Indication struct {

   Detector *Detector `json:"detector" validate:"required"`

   DangerLevel DangerLevel `json:"dangerLevel" validate:"required"`

   P10 int64 `json:"p10" validate:"required"`

   P25 int64 `json:"p25" validate:"required"`

   NormalP10 int64 `json:"normalP10" validate:"required"`

   NormalP25 int64 `json:"normalP25" validate:"required"`

   Pressure float64 `json:"pressure" validate:"required"`

   WindSpeed float64 `json:"windSpeed" validate:"required"`

   Humidity float64 `json:"humidity" validate:"required"`

   Temperature float64 `json:"temperature" validate:"required"`
}

func NewIndication(
    detector *Detector,
    dangerLevel DangerLevel,
    p10 int64,
    p25 int64,
    normalP10 int64,
    normalP25 int64,
    pressure float64,
    windSpeed float64,
    humidity float64,
    temperature float64,
) *Indication {
    return &Indication{
        Detector: detector,
        DangerLevel: dangerLevel,
        P10: p10,
        P25: p25,
        NormalP10: normalP10,
        NormalP25: normalP25,
        Pressure: pressure,
        WindSpeed: windSpeed,
        Humidity: humidity,
```

```
        Temperature: temperature,
    }
}

type DetectorsRepo interface {
   GetRecentIndications(l zerolog.Logger) ([]*client.Indication, error)
}

func (cl Handlers) MountDetectors(r *routing.Mux, repo DetectorsRepo) {
   r.MethodPath(cl.GetRecentIndicationsHandler(func(call
*client.GetRecentIndicationsCall) {
      l := cl.LoggerFor(call)

      l.Info().Msg("begin handling")

      indications, err := repo.GetRecentIndications(l)
      if err != nil {
         web.RespondError(call, err)
         return
      }

      l.Info().Msg("finish handling")

      call.RespondOk(indications)
   }))
}
```

Ниже представлен модуль работы с базой данных для получения показаний:

```
func NewDetectorsRepo(db *dbms.DB) *DetectorsRepo {
   return &DetectorsRepo{
      DB:     db,
      Client: &http.Client{},
   }
}

type DetectorsRepo struct {
   *dbms.DB
   *http.Client
}

func (d *DetectorsRepo) GetRecentIndications(l zerolog.Logger)
([]*client.Indication, error) {
```

```
    req, err := http.NewRequest("GET", "https://ecoradar.online/api.php", nil)
    if err != nil {
        l.Error().Err(err).Msg("can not create infinnity request")
        return nil, err
    }
    req.Header.Add("Authorization", "EC803463-E492-4BF7-B0F7-A962CA7373FA")

    resp, err := d.Do(req)
    if err != nil {
        l.Error().Err(err).Msg("can not make infinnity request")
        return nil, err
    }

    var detectors []*detector
    err = json.NewDecoder(resp.Body).Decode(&detectors)
    if err != nil {
        l.Error().Err(err).Msg("can not decode infinnity detectors")
        return nil, err
    }

    // Сначала в этот список будут помещены все
 сохранённые датчики,
    // а потом будут удалены все актуальные
    var detectorIdsToRemove []int64
    err = d.QueryRow(`
SELECT ARRAY(
     SELECT id
     FROM detectors)`,
    ).Scan(
        d.Array(&detectorIdsToRemove),
    )
    if err != nil {
        l.Error().Err(err).Msg("can not select detector ids to remove")
        return nil, err
    }

    indications := make([]*client.Indication, 0)
    for _, detector := range detectors {
        detectorIdsToRemove = remove(detectorIdsToRemove, detector.Id)

        var dangerLevel client.DangerLevel
        if detector.P10 > 50 || detector.P25 > 25 {
            dangerLevel = client.DangerLevelHigh
        } else {
```

```
            if detector.P10 > 25 || detector.P25 > 12 {
                dangerLevel = client.DangerLevelMedium
            } else {
                dangerLevel = client.DangerLevelLow
            }
        }

        _, err = d.Exec(`
INSERT INTO detectors (id)
VALUES ($1)
ON CONFLICT (id)
    DO UPDATE SET id = $1`,
            detector.Id,
        )
        if err != nil {
            l.Error().Err(err).Msg("can not insert detector")
            return nil, err
        }

        indication := client.NewIndication(
            client.NewDetector(
                detector.Id,
                client.NewLocation(detector.Lat, detector.Lon),
            ),
            dangerLevel,
            detector.P10,
            detector.P25,
            50,
            25,
            detector.Pressure,
            detector.WindSpeed,
            detector.Humidity,
            detector.Temperature,
        )

        indications = append(indications, indication)
    }

    _, err = d.Exec(`
DELETE FROM detectors
WHERE id = ANY($1)`,
        d.Array(detectorIdsToRemove),
    )
    if err != nil {
        l.Error().Err(err).Msg("can not delete odd detectors")
```

```
        return nil, err
    }

    return indications, nil
}

type detector struct {
    Id          int64   `json:"eco_id"`
    Lat         float64 `json:"eco_lat"`
    Lon         float64 `json:"eco_lon"`
    P10         int64   `json:"eco_sds_p1"`
    P25         int64   `json:"eco_sds_p2"`
    Pressure    float64 `json:"eco_pressure"`
    Temperature float64 `json:"eco_ds18b20_temperature"`
    WindSpeed   float64 `json:"eco_wind_speed"`
    Humidity    float64 `json:"eco_humidity"`
}

func remove(l []int64, item int64) []int64 {
    for i, other := range l {
        if other == item {
            return append(l[:i], l[i+1:]...)
        }
    }
    return l
}
```

В коде бизнес-логики ни разу не упоминаются сущности из модуля управления базой данных. Это означает, что правило зависимостей соблюдается для разработанного приложения.

При разработке остальных модулей приложения были применены аналогичные принципы проектирования.

### 2.7 Мобильное приложение

Было разработано мобильное приложение на языке Dart.

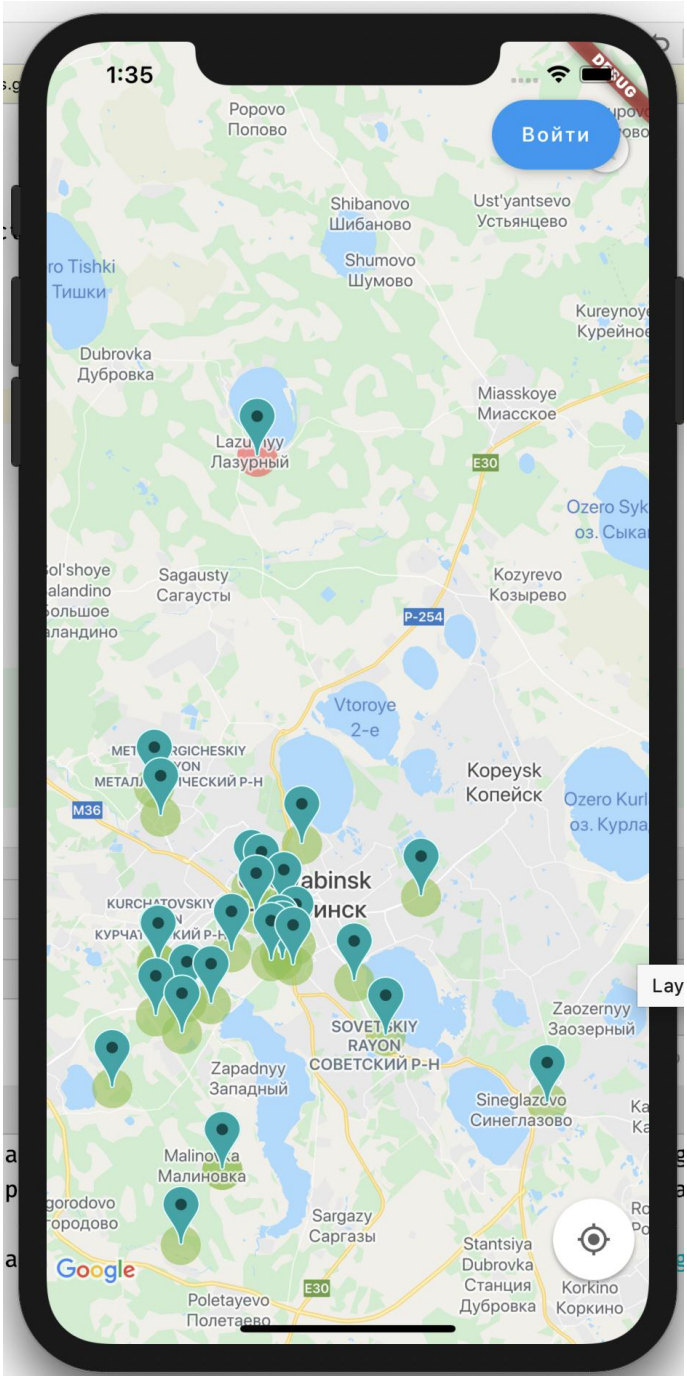


Рисунок 23. Мобильное приложение Ecoradar.online

Пользователь может получить детальную информацию о содержании микродисперсных частиц, если это необходимо.

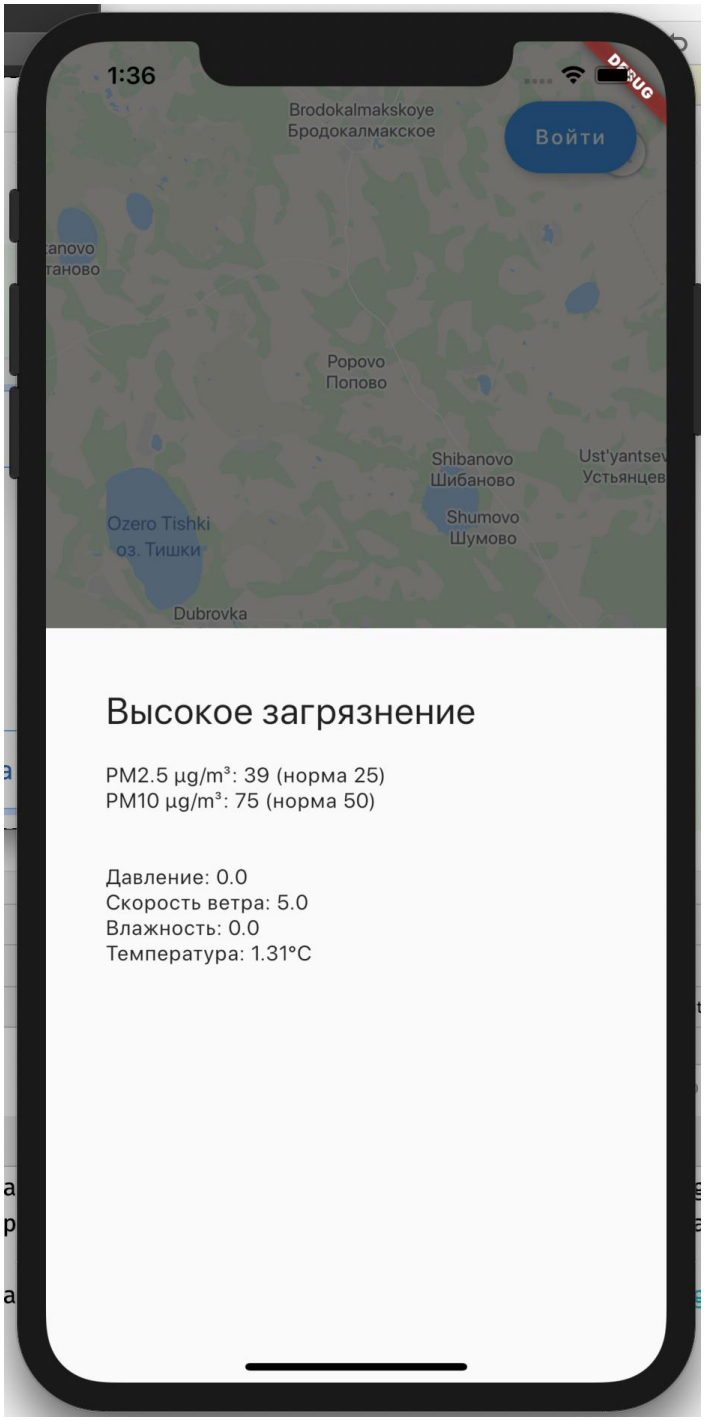


Рисунок 24. Детальная информация о показании

Так же у пользователя есть возможность авторизироваться в приложении с помощью универсального аккаунта компании Google.

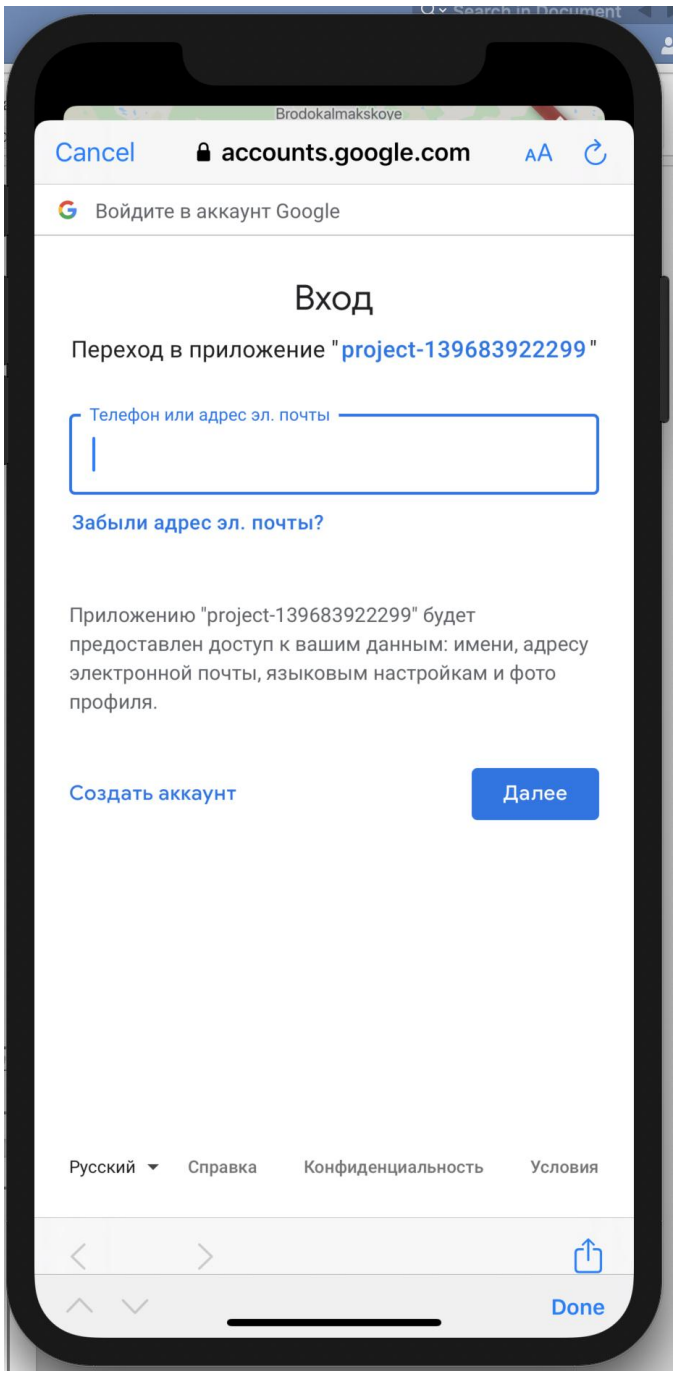


Рисунок 25. Авторизация через Google

После авторизации, пользователь может отслеживать показания интересных ему датчиков. Ему будут приходить уведомления в случае изменения уровня загрязнения на этих датчиках.

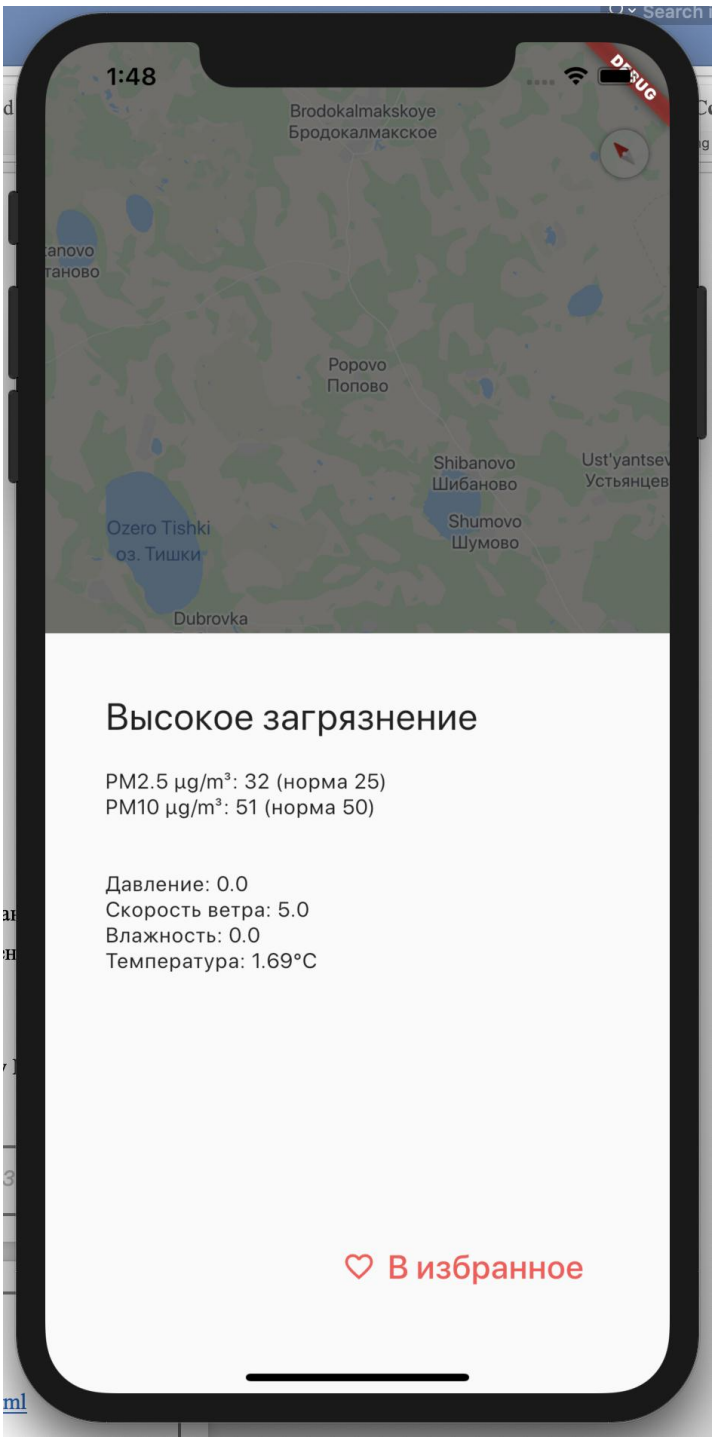


Рисунок 26. Добавление в «избранное»

**2.8 Вывод:** были проанализированы требования к информационной системе, выделены прецеденты, проанализированы возможные технологии реализации и выполнена реализация мобильного приложения

# ГЛАВА 3. ОЦЕНКА ЭФФЕКТИВНОСТИ ПРОЕКТА

## 3.1 Составление перечня работ

Для реализации данного проекта необходимо выполнить работы, перечисленные в таблице 10.

| Наименование этапа разработки | Исполнители (должность) | Тродоёмкость, час | Продолжительность, дней |
|---|---|---|---|
| 1. Подготовительный этап | | | |
| 1.1 Анализ информационной составляющей проекта | Бизнес-аналитик, руководитель проекта | 24 | 3 |
| 1.2 Анализ внешнего API сервиса хранения и сбора данных | Архитектор | 8 | 1 |
| 1.3 Анализ требований к системе | Бизнес-аналитик, руководитель проекта | 40 | 5 |
| 1.4 Описание требований в виде прецедентов | Бизнес-аналитик | 24 | 3 |
| 1.5 Составление диаграммы прецедентов | Бизнес-аналитик | 8 | 1 |
| 1.6 Анализ технических требований к системе | Архитектор, руководитель проекта | 16 | 2 |
| 1.7 Выбор | Архитектор | 8 | 1 |

| | | | |
|---|---|---|---|
| подходящего технологического стека | | | |
| 1.8 Описание технологического стека | Архитектор | 8 | 1 |
| Итого | | 136 | 17 |
| 2. Основной этап | | | |
| 2.1 Проектирование архитектуры системы | Архитектор | 24 | 3 |
| 2.2 Описание API серверной части | Архитектор | 16 | 2 |
| 2.3 Разработка серверной части | Серверный программист | 80 | 10 |
| 2.4 Разработка клиентской части | Мобильный программист | 80 | 10 |
| Итого | | 200 | 25 |
| 3. Закупка необходимых технических средств | | | |
| 3.1 Установка сервера | Серверный программист | 4 | 0.5 |
| 3.2 Развертка серверной части | Серверный программист | 4 | 0.5 |
| Итого | | 8 | 1 |
| 4. Заключительный этап | | | |

| | | | |
|---|---|---|---|
| 4.1 Сбор аналитических данных по системе | Бизнес-аналитик | 8 | 1 |
| 4.2 Составление отчётности по реализации системы | Руководитель проекта | 8 | 1 |
| Итого | | 16 | 2 |
| Всего | | 360 | 45 |

### 3.2 Составление перечня необходимых ресурсов

Для осуществления контроля за разработкой программного продукта будет назначен руководитель проекта из числа специалистов компании (с окладом 50 тыс. руб.). Результаты расчетов расходов представлены в таблице 11.

Таблица 11 – Расходы на управление

| | Ставка | Стоимость чел/часа | | Загрузка | | Прод., мес. | Выработка, руб. |
|---|---|---|---|---|---|---|---|
| Руководитель проекта | 50000 | 312 | руб. | 44 | ч/месяц | 2 | 27456 |

Расходы на разработку информационной системы, а так же расходы на аналитику, проектирование развёртку включают в себя стоимость времени работы специалистов. Ставка серверного программиста составляет 60 тысяч рублей, ставка мобильного программиста составляет 50 тысяч рублей, ставка архитектора составляет 100 тысяч рублей, ставка бизнес-аналитика составляет 60 тысяч рублей.

Таблица 12 – Расходы на управление

| | Ставка | Стоимость чел/часа | | Загрузка | | Прод., мес. | Выработка, руб. |
|---|---|---|---|---|---|---|---|
| Архитектор | 100000 | 625 | руб. | 40 | ч/месяц | 2 | 50000 |
| Бизнес-аналитик | 60000 | 375 | руб. | 52 | ч/месяц | 2 | 39000 |
| Серверный программист | 60000 | 375 | руб. | 80 | ч/месяц | 1 | 30000 |
| Мобильный программист | 50000 | 312 | руб. | 80 | ч/месяц | 1 | 24960 |

Общие расходы на оплату выработанных часов составили 171416 руб.

Дополнительной статьёй расходов является установка сервера. Его стоимость составила 30000 руб.

Исходя из расчётов, общие расходы на проект составят 201416 руб.

### 3.3 Рассчёт прогнозируемого эффекта на выполнение стратегических целей компании

На стратегической карте (таблица 1) устанавливаются целевые показатели, представляющие собой конкретные значения, которые стремится достичь организация. В таблице 13 представлено прогнозируемое влияние на стратегические цели компании.

Таблица 13. Прогнозируемое влияние на стратегические цели компании

| Цель | Показатель | Результат к 2023 году | Прогнозируемый результат Ecoradar |
|---|---|---|---|
| Увеличение количества упоминаний в СМИ | Количество упоминаний в региональных СМИ | +10 | +5 |
| | Количество упоминаний в федеральных СМИ | +5 | +3 |

| | | | |
|---|---|---|---|
| Увеличение количества скачиваний приложений | Количество скачиваний приложений | +20 000 | +20 000 |
| Увеличение количества положительных отзывов в социальных сетях | Количество положительных отзывов в социальных сетях | +50 | +30 |

**3.4 Вывод:** исходя из прогнозируемых показателей от внедрения системы, можно сделать вывод о том, что она окажет значительный эффект на перспективу узнаваемости и медийной известности компании.

**Заключение**

В Челябинске в последние годы проблема загрязнения окружающей среды становится всё более и более актуальной. Существуют различные источники информации по этой теме, но, они, по большей части, не предоставляют структурированной информации.

В рамках проекта была описана структура и виды деятельности компании ООО «Infinnity Solutions», был проведён анализ её ближнего и дальнего окружения, определены основные проблемы отображения данных с датчиков, а так же, предложено их решение.

Для решения этих проблем была разработана информационная система, которая позволяет пользователю получать информацию о загрязнении окружающей среды в городе Челябинске в реальном времени.

Система предоставляет как графический интерфейс для просмотра уровня загрязнения, полученного с датчиков, так и уведомления об изменении уровня загрязнения на тех датчиках, которые пользователь добавил в «избранное».

С помощью данной системы, может быть значительно повышен уровень узнаваемости компании «Infinnity», что поможет компании достигнуть установленных целей.

Таким образом, все поставленные на начальных этапах работы задачи решены, цель работы в полной мере достигнута.

**Библиографический список**

[1] Сайт компании «Infinnity Solutions» [Электронный ресурс] – Режим доступа: https://infinnity.ru/

[2] Арутюнова Д.В. Стратегический менеджмент

[3] Особенности формирования счетной карты [Электронный ресурс] – Режим доступа: https://rich-c.ru/osobennosti-formirovaniya-schetnoy

[4] Импортозамещение программного обеспечения в госсекторе [Электронный ресурс] – Режим доступа: http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%98%D0%BC%D0%BF%D0%BE%D1%80%D1%82%D0%BE%D0%B7%D0%B0%D0%BC%D0%B5%D1%89%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2_%D0%B3%D0%BE%D1%81%D1%81%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%B5

[5] Борисов С.А. Определение «информационной интенсивности» организации с использованием матрицы МакФарлана

[6] Ларман, Крэг. Применение UML 2.0 и шаблонов проектирования. Практическое руководство. 3-е издание. Пер. с англ. – М.: ООО «И.Д. Вильямс», 2009

[7] Richard Jones, Antony Hosking, Eliot Moss. The Garbage Collection Handbook. The Art of Automatic Memory Management, стр. 23

[8] Jonh Ousterhout, A Philosohpy of Software Design, глава 4

[9] Документация языка Go [Электронный ресурс] – Режим доступа: https://dlintw.github.io/gobyexample/public/memory-and-sizeof.html

[10] How to Get the Size of an Object in Java [Электронный ресурс] – Режим доступа: https://www.baeldung.com/java-size-of-object

[11] Документация платформы .NET [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-3.5/bb384547(v=vs.90)?redirectedfrom=MSDN

[12] Документация языка Python [Электронный ресурс] – Режим доступа: https://docs.python.org/3/library/sys.html#sys.getsizeof

[13] The Java Virtual Machine [Электронный ресурс] – Режим доступа: https://www.artima.com/insidejvm/ed2/jvmP.html

[14] JEP 254: Compact Strings [Электронный ресурс] – Режим доступа: https://openjdk.java.net/jeps/254

[15] Go versus Java fastest programs [Электронный ресурс] – Режим доступа: https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go.html

[16] Go versus Python3 fastest programs [Электронный ресурс] – Режим доступа: https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go-python3.html

[17] C# .NET Core versus Java fastest programs [Электронный ресурс] – Режим доступа: https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/csharp.html

[18] Memory and IO in Database Performance [Электронный ресурс] – Режим доступа: http://www.qdpma.com/SystemArchitecture/MemoryIO.html

[19] Martin, Robert. The Clean Architecture [Электронный ресурс] – Режим доступа: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

[20] Википедия, Принцип инверсии зависимости [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF_%D0%B8%D0%BD%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B8_%D0%B7%D0%B0%D0%B2%D0%B8%D1%81%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B5%D0%B9

[21] Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения, стр. 74

[22] Лицензия СУБД «PostgreSQL» [Электронный ресурс] – Режим доступа: https://github.com/postgres/postgres/blob/master/COPYRIGHT

[23] Поддерживаемые СУБД «PostgreSQL» стандарты [Электронный ресурс] – Режим доступа: http://www.lghost.ru/docs/postgres/postgres/features.html

[24] Поддерживаемые СУБД «MySQL» стандарты [Электронный ресурс] – Режим доступа: https://dev.mysql.com/doc/refman/8.0/en/compatibility.html

[25] Импортозамещение программного обеспечения в госсекторе [Электронный ресурс] – Режим доступа: http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%98%D0%BC%D0%BF%D0%BE%D1%80%D1%82%D0%BE%D0%B7%D0%B0%D0%BC%D0%B5%D1%89%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2_%D0%B3%D0%BE%D1%81%D1%81%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%B5

[26] Преимущества использования языка plSQL [Электронный ресурс] – Режим доступа: https://www.postgresql.org/docs/current/plpgsql-overview.html#PLPGSQL-ADVANTAGES

[27] Документация языка Go [Электронный ресурс] – Режим доступа: https://golang.org/pkg/database/sql/

[28] Документация языка Python [Электронный ресурс] – Режим доступа: https://www.python.org/dev/peps/pep-0249/

[29] Документация технологии «JDBC» [Электронный ресурс] – Режим доступа: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/index.html

[30] Документация платформы «.NET» [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/data-providers

[31] Документация технологии «SqlAlchemy» [Электронный ресурс] – Режим доступа: https://docs.sqlalchemy.org/en/13/

[32] Документация технологии «JOOQ» [Электронный ресурс] – Режим доступа: https://www.jooq.org/javadoc/latest/org.jooq/module-summary.html
[33] Документация технологии «Linq» [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/linq/
[34] Переиспользование кодовой базы между iOS, Android и Web [Электронный ресурс] – Режим доступа: https://medium.com/@john.p.ryan4/sharing-code-on-ios-android-and-web-85e8ed7dfccd
[35] Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения, стр. 155
[35] Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения, стр. 161