ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Программная инженерия» профессор
департамента программной
инженерии, канд. техн. наук

_____ В. В. Шилов
«_____» _____ 2021 г.

Выпускная квалификационная работа на тему

«CRM-система для благотворительного фонда «AIAIN». Web-приложение для сотрудников фонда»

по направлению подготовки 09.03.04 «Программная инженерия»

Доцент департамента
образовательной программы
«Программная инженерия»

Х. М. Салех

2.06.2021 _____
(дата, подпись)

Выполнила студентка
образовательной программы
09.03.04 «Программная инженерия»
группы БПИ 174
Д. Ю. Редникина

2.06.2021 _____
(дата, подпись)

Москва 2021

Реферат

Данная работа посвящена проектированию и разработке Web-приложения для сотрудников благотворительного фонда «AIAIN». Деятельность фонда направлена на сбор пожертвований и распределение собранных средств на нужды людей. Разработанное Web-приложение позволит сотрудникам фонда автоматизировать бизнес процессы работы с поступающими заявками на пожертвования, а также коммуникации с пользователями посредством реализации следующего функционала:

– Создание панели администрирования пользователей системы;

– Автоматизация бизнес процесса обработки заявок на пожертвование;

– Создание чатов поддержки для связи с пользователями мобильного приложения;

– Мониторинг и занесение в систему пожертвований;

– Просмотр аналитики и многое другое;

Данная работа является частью группового проекта по разработке CRM-системы для благотворительного фонда «AIAIN», которая включает в себя также мобильное приложение для доноров и нуждающихся, а также общий сервер. При разработке Web-клиента для сотрудников фонда применялись следующие технологии: язык разработки Typescript, Elm, фреймворк - React.

Работа содержит 57 страниц, 24 рисунков, 10 таблиц, 4 листинга, 27 источников.

Ключевые слова – работники фонда; благотворительность; НКО; пожертвование; web;

Abstract

The proposed work is devoted to the devepoment of the Web-application for the charitable foundation «AIAIN». Fund's key activities are collecting donations and distributing collected money for the people's needs. Developed Web-application will allow charity stuff to automate business processes of working on incoming applications, and also will help to communicate directly with donors and donees. The following functionality is done to help automate mentioned workflows:

– Administration of the CRM users;

– Automation of the business process of working on incoming application;

– Chat support for communication with mobile application users (donors and donees);

– Monitoring of incoming donations;

– Analytics and more;

The following work is a part of the team project called «CRM-System for Charitable Foundation «AIAIN», which includes mobile application for donors and donees and backend. The following technologies were used for the Web-application development: language Typescript, Elm, framework - React.

The work contains 57 pages, 24 diagrams, 10 tables, 4 listings, 27 sources.

Keywords – fund employees; charity; Charitable Foundation; donation; web;

Основные определения, термины и сокращения

CRM-система – это прикладное программное обеспечение для организаций, предназначенное для автоматизации стратегий взаимодействия с заказчиками (клиентами), в частности, для повышения уровня продаж, оптимизации маркетинга и улучшения обслуживания клиентов путем сохранения информации о клиентах и истории взаимоотношений с ними, установления и улучшения бизнес-процессов и последующего анализа результатов.

Донор – человек, безвозместно жертвующий денежную сумму другому человеку.

Благотворительный фонд – некоммерческая организация, учреждённая гражданами и (или) юридическими лицами на основе добровольных имущественных взносов, преследующая благотворительные, культурные, образовательные или иные социальные, общественно полезные цели.

Некоммерческая организация – организация, не имеющая в качестве основной цели своей деятельности извлечения прибыли. Некоммерческие организации могут создаваться для достижения социальных, благотворительных, культурных, образовательных, политических, научных и управленческих целей

Сотрудники фонда – сотрудники фонда могут иметь следующие роли в системе: администратор; менеджер; член комиссии; контент-менеджер; оператор;

Закят – один из пяти столпов ислама, обязательный ежегодный налог в исламском праве, выплачиваемый с различного вида доходов и имущества (движимого и недвижимого) всеми самостоятельными, свободными, дееспособными и взрослыми мусульманами в пользу нуждающихся единоверцев. Согласно шариату, его выплата означает, что полученные доходы и нажитое богатство не являются греховными.

Содержание

Введение

Статистика показывает [1], что количество пользователей веб-сайтов значительно превосходит популярность мобильных приложений. Кроме того, интернет сайты не нужно адаптировать под разные платформы/операционные системы в отличие от мобильных или десктоп приложений. Это связано с тем, что веб-сайтам достаточно того, чтобы веб-браузеры поддерживали JavaScript (во все популярные веб-браузеры такая поддержка встроена по умолчанию).

Что касается предметной области благотворительности, то с каждым годом количество доноров увеличивается [2] и возникает потребность автоматизированно обрабатывать поступающие пожертвования, вести статистику доноров и многое другое. Большинство фондов до сих пор ведут учет данных в электронных таблицах, но это трудоемкий и ресурсозатратный процесс.

Программное обеспечение для управления отношениями с учредителями (CRM) - это способ для некоммерческих организаций отслеживать свои отношения с донорами и нуждающимися. CRM-система регистрирует прошлое участие и основные данные о донорах, помогает некоммерческим организациям реализовывать индивидуальные стратегии охвата и дает организациям понимание с помощью надежных отчетов.

К сожелению, большинство существующих решений не учитывают особенности предметной области и специфичные бизнес процессы фондов. Поэтому в рамках командного проекта была разработана CRM-система для благотворительного фонда «AIAIN», которая учитывает все особенности работы фонда. Именно для такой платформы разрабатывается Web-приложение, которое будет визуализировать аналитические отчеты, позволит сотрудникам фонда управлять заявками и администрировать пользователей.

Данная работа посвящена разработке веб-приложения для благотворительного фонда «AIAIN». Целью работы является создание клиентского Web-приложения. Для достижений этой цели требуется решить следующие задачи:

- Провести анализ существующих решений;
- Собрать требования от заказчика и ключевых стейкхолдеров;
- Разработать макеты с помощью инструмента Figma;
- Выбрать технологии, фреймворки, библиотеки;
- Реализовать компоненты по разработанным макетам;
- Реализовать слой бизнес логики согласно требованиям;
- Настроить взаимодействие с API;
- Произвести тестирование продукта;
- Подготовить техническую документацию;

В первой главе рассматриваются текущие технологии и решения для создания Web-приложений для CRM-системы. Во второй и третьей главах показаны архитектурные особенности реализации приложения и основные технологические подходы.

Глава 1. Обзор предметной области

1.1 Рамки проекта

Ежегодно сотни тысяч людей жертвуют свои деньги некоммерческим организациям. Еще больше людей обращаются за помощью в общественные благотворительные фонды. Управлять благотворительным фондом становится все сложнее. Нужно не только принимать пожертвования и регистрировать заявки на сборы средств, но и контролировать работников фонда, волонтеров, подготавливать документацию для вышестоящих органов и многое другое.

Следовательно, некоммерческие организации нуждаются в платформе с функционалом, ориентированным на бизнес процессы фонда, чтобы обрабатывать всю необходимую информацию в одном месте. К сожалению, большинство фондов пользуются электронными таблицами или, что еще хуже, ведут записи в бумажной форме. Как результат, на каждое действие тратятся большое количество времени и ресурсов. Арабский фонд «AIAIN» также столкнулся с проблемой автоматизации бизнес процессов, специфичных для предметной области благотворительности.

Разрабатываемое Web-приложение ориентировано на специфические нужды некоммерческой организации «AIAIN». Web-приложение для сотрудников фонда может предоставить все преимущества, которыми бизнес-компании пользовались в течение многих лет и чего так не хватало этому фонду. Это также позволит наладить бизнес-процессы внутри фонда «AIAIN», настроить тайм-менеджмент и автоматизировать составление отчетности.

1.2 Проблема

Арабский фонд «AIAIN», расположенный в Объединенных Арабских Эмиратах, нуждается в платформе, объединяющей бизнес-процессы фонда в одном месте. В данный момент фонд пользуется несколькими сторонними системами для обработки заявок, ведения учета пожертвований, сбора статистики, коммуникации с донорами и нуждающимися – на данные действия сотрудники фонда тратят большое количество времени и усилий.

На рассмотрение и принятие одной заявки сотрудники фонда «AIAIN» тратят от 1 недели до 1 месяца, так как процесс включает в себя рассмотрение заявки несколькими членами комиссии и возможной доработки файлов и материалов от нуждающегося. Вся коммуникация как между сотрудниками фонда, так и между менеджерами и нуждающимися происходит по разным каналам связи, из-за этого часто информация теряется, устаревает и ее потом сложно найти и отследить.

Также при принятии заявок происходит голосование членов комиссии, ответственных за ту или иную категорию заявки. Данный процесс также никак не автоматизирован и не документируется - это плохо сказывается на отслеживании истории принятии заявок.

Таким образом, были выделены следующие актуальные проблемы фонда «AIAIN»:

– Сотрудники не пользуются централизованной системой, в которой бы находились все процессы по обработке заявок, общению с донорами и т.д;

– Отсутствует автоматическая система оповещения при изменении статуса решения по какой-то заявке;

– Очень много времени тратится на обработку и принятие решения по одобрению заявки на пожертвование;

– Учет поступающих пожертвований не автоматизирован и ведется вручную;

– Отсутствует подходящая под нужды фонда система, которая соответствовала бы особенностям организации ОАЭ и законодательству;

1.3 Роль в команде

Работа выполняется в рамках командного проекта «CRM-система для благотворительного фонда «AIAIN». В систему входят следующие части:

– Android-приложение для размещения и отслеживания пожертвований, разработано Анной Михалевой в рамках ВКР;

– Backend-приложение, разработанное Контантином Манежиным в рамках ВКР;

– Blockchain-модуль, разработанный Костюченко Ильей в рамках проектной работы на 4 курсе;

В рамках данной выпускной квалификационной работы была разработана часть Web-приложение для сотрудников фонда, пример взаимодействия всех 4 частей представлен на диаграмме 1.

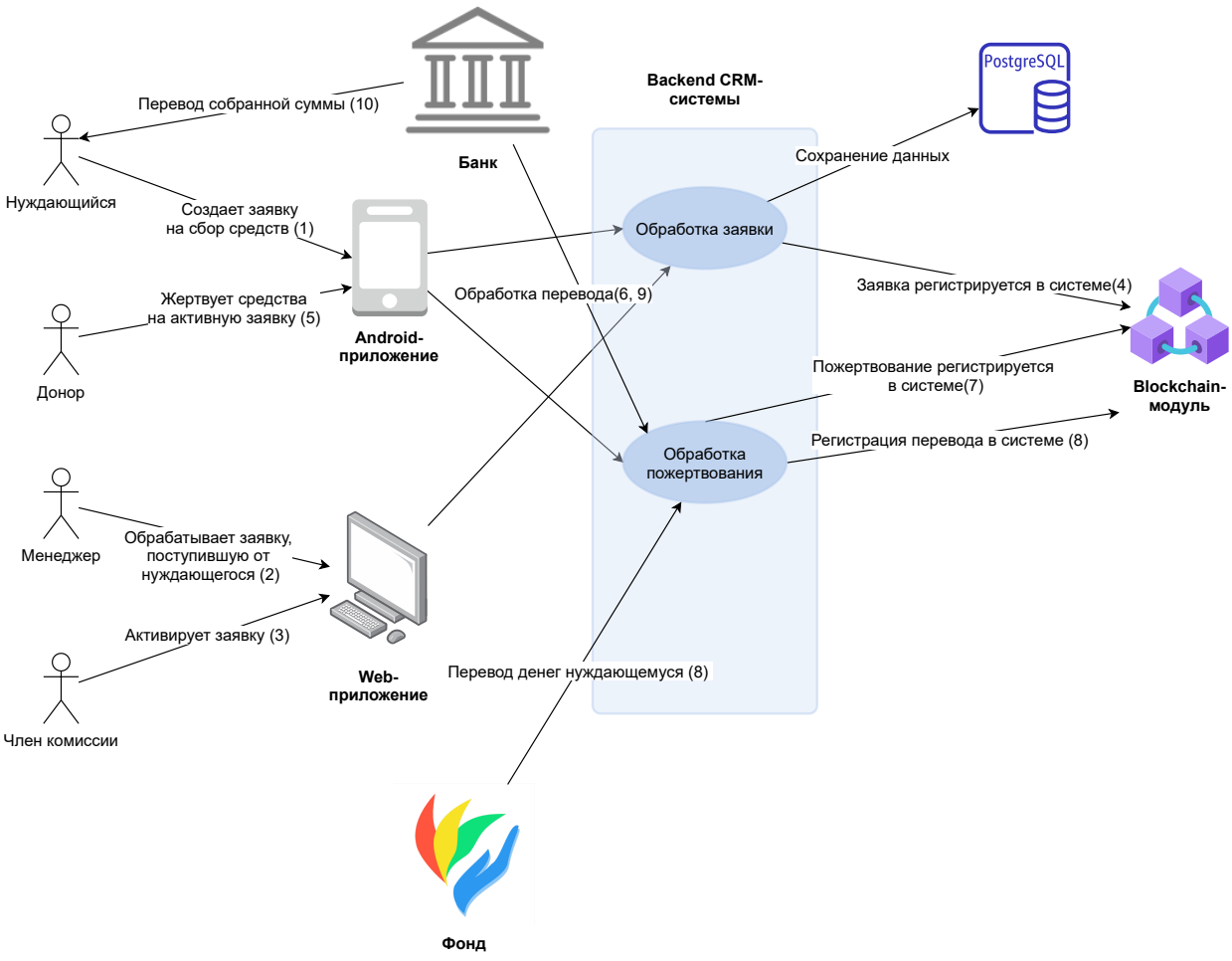

Рисунок 1 — Взаимодействие частей CRM-системы

1.4 Потенциальные пользователи и заинтересованные лица

1.4.1 Устройство рынка

В работе благотворительного фонда задействовано много людей - различные сотрудники фонда, которые обрабатывают поступающие средства и заявки, руководство фонда, бухгалтерия и многие другие. Так как сейчас популярность благотворительных организаций растет с каждым годом, то работа с массовым жертвователем требует отлаженных бизнес-процессов, утверждают эксперты [3].

Прежде чем открывать сбор средств, фонды проводят экспертизу заявок от потенциальных получателей помощи. Этот процесс трудоемкий и требует большого количества ресурсов, так как заявки приходят ручной отбор.

Также стоит затронуть бизнес процесс учета пожертвований. Самый важный для потенциального жертвователя вопрос – могут ли фонды гарантировать, что деньги потрачены с пользой? Пока единого решения для мониторинга поступающих средств и входящих заявок не существует - большинство фондов ведут учет данных в электронных таблицах.

Существуют решения в виде CRM-систем для объединенния учета пожертвтований и заявок. Данные решения доступны по подписке и обычно предоставляют лишь ограниченный фукнционал, который покрывает все бизнес-процессы, специфичные для фонда «AIAIN».

Таким образом - главное заинтересованное лицо, благотворительная организация «AIAIN» не нашла подходящего решения под свои нужды - администрирование персонала, автоматизированное управление и учет заявок и пожертвований, удобная отчетность и аналитика.

1.4.2 Пользовательская среда

Разрабатываемый продукт будет применяться сотрудниками фонда «AIAIN» в рабочих целях. Продуктом смогут пользоваться как администраторы, менеджеры фонда, так и комиссия по принятию решений.

Задачи, которые решают пользователи данной системой давольно обширные. Это может быть как обработка заявки и сбор необходимых данных от пользователя, так и более крупные задачи:

- Администрирование пользователей системы;
- Поддержка пользователей в чате;
- Мониторинг и анализ статистики фонда;
- Размещение контента о фонде;
- Сбор информации о совершенных пожертвованиях, активных донорах фонда;

Перечисленные задачи являются специфичными для предметной области благотворительности и на данный момент фонд их решает с использованием различных систем. Это очень неудобно, так как системы не агрегированы между собой и нет удобного интерфейса для сотрудников фонда.

1.4.3 Список пользователей

В нижеприведенной таблице 9 можно увидеть описание будущих пользователей системы. Всего сотрудников фонда «AIAIN» в штате 12 человек.

Таблица 1 — Список пользователей

Роль	Описание	Способ работы
Администратор	Сотрудник фонда, задача которого упарвлять пользователями системы, а также мониторить логи. Возраст 20-70 лет. Работает на 1/2 ставку.	Управляет всеми учетными записями в системе. У него есть доступ к логам системы, а также возможность просматривать пользователей системы и регистрировать их.
Оператор	Является сотрудником службы поддержки доноров и нуждающихся. Возраст 20-70 лет. Работает на 1/2 ставку.	Отвечает на сообщения пользователей, отправленные через чат.
Контент-менеджер	Отвечает за контент, размещаемый на основной странице фонда. Возраст 20-70 лет. Работает на 1/2 ставку.	Имеет возможность редактировать описание фонда, а также раздел часто задаваемых вопросов и новости фонда.
Менеджер	Обрабатывает заявки, поступающие в фонд. Возраст 20-70 лет. Полная занятость.	Имеет доступ к заявкам фонда, а также может обрабатывать данные заявки: менять статусы, редактировать и общаться с пользователем в рамках одной заявки. Также менеджер может просматривать транзакции фонда.
Член комиссии	Члены комиссии фонда принимают решение об опубликовании заявок. Возраст 20-70 лет. Полная занятость.	Имеет доступ к расширенному функционалу управления заявками, а именно к активации заявки (активная заявка становится видимой для всех пользователей системы, на нее можно жертвовать деньги).

1.5 Анализ существующих аналогов

На этапе анализа необходимо сделать обзор существующих решений, а именно - выделить прямых конкурентов, выявить их сильные стороны и сравнить с разрабатываемым Web-приложением.

1.5.1 Краткое описание существующих решений

При поиске существующих решений была составлена таблица 2 с кратким описанием назначения решения и определением типа конкурента [4]. При детальном рассмотрении и дальнейшем подробном анализе были выбраны только прямые конкуренты. Жирным

шрифтом в таблице 2 выделены конкуренты, функциональность которых наиболее пересекается с требованиями, предоставленными заказчиком. Они будут рассмотрены детально далее.

Таблица 2 — Виды конкурентов

Название	**Описание**	**Вид**
Donorfy	CRM для сбора средств	Прямой
DonorPerfect	Программное обеспечение для сбора средств онлайн	Прямой
Beacon	CRM для благотворительных организаций	Прямой
Access thankq	CRM для благотворительных организаций	Прямой
Harlequin charity CRM	CRM для благотворительных организаций	Прямой
Copper	CRM-система общего назначения	Косвенный
Virtuous	CRM-система, ориентированная на пожертвования и маркетинг	Прямой
Salesforce.com	CRM-система с настраеваемым функционалом	Косвенный
eTapestry	CRM-система для НКО	Прямой
Raiser's edge	CRM-система для НКО	Прямой
CiviCRM	CRM-система для НКО	Прямой
monday.com	CRM платформа общего назначения	Косвенный
Google Sheets	Электронные таблицы	Косвенный
1C	CRM-система	Косвенный
Microsoft Dynamics 365	CRM-система	Косвенный
Instagram	Социальная сеть	Косвенный
Facebook Fundraisers	Социальная сеть	Косвенный

Donorfy[1]

CRM-система для благотоврительный организаций. Используется более чем в 15 фондах, большинство из которых находятся в Дании. Возможна кастомизация функционала, ориентированная на специфику бизнес процессов. Есть интеграция с популярными mailing-сервисами, сервисами по бронированию билетов, PayPal и т.д. Предоставляют функциональные аналитические панели. Поддерживают только три типа пользователя: Administrator, Standard, Viewer. Есть возможность подключить примем пожертвований в виде виджета на веб-сайте. Есть возможность просмотреть поступающие пожертвования от доноров.

Beacon[2]

CRM-система для НКО, в рейтинге систем для благотворительных организаций за-

[1] `https://donorfy.com/features`

[2] `https://www.beaconcrm.org`

няла 1 место в 2020 году [5]. Предоставляет организациям 14 дней пробного периода, после - оплата каждый месяц. Есть дашборды для сотрудников с задачами. Поддерживает добавление организаций в справочник. Сервис берет комисиию при совершении оплаты через форму на их сайте. Нет поддержки таких ролей как администратор, контент менеджер и оператор.

Access thankq[3]

CRM-система для НКО из Великобритании, более 10000+ клиентов по всему миру. Функциональность направлена на управление встречами, событиями, а также на мониторинг доноров и волонтеров фонда. Как и предыдущие конкуренты, есть возможность интеграции с популярными платежными сервисами. Предоставляет аналитические панели для мониторинга системы.

Harlequin charity CRM[4]

CRM-система, разработанная в Великобритании. Функциональность ориентирована на небольшие НКО для работы с финансами, а также мониторинг доноров и волонтеров. Только платный функционал. Предлагает настраеваемый интерфейс и функционал.

DonorPerfect[5]

CRM система специально для НКО. Имеется мобильное приложение и поддержка пожертвований. Поддерживает ролевую модель сотрудников фонда. Предоставляет только платный функционал, есть аналитические панели.

1.5.2 Анализ конкурентов

Взяв за основу критерии (см. изображение 2) из проведенного исследования [5] популярных CRM-систем была составлена аналогичная таблица с критериями и метриками, важными в предметной области благотворительности и пожертвований.

Software	Number of responses	Functionality	Cost	Integration with other systems	Integration with website	Ease of use	Accessibility	Security	Ability to customise	Overall
Access thankQ	7	3.4	2.9	2.9	2.4	3.6	3.6	3.7	2.4	3.0
Beacon	27	4.9	4.7	4.7	4.8	4.9	4.9	4.7	4.9	4.9
Donorfy	72	4.5	4.6	4.4	4.2	4.5	4.6	4.5	4.0	4.6
eTapestry (Blackbaud)	8	3.1	2.9	3.1	2.8	2.6	3.4	4.1	3.1	3.0
Harlequin	13	4.1	4.0	3.4	3.0	4.0	4.1	4.1	4.0	3.9
Raiser's Edge (Blackbaud)	39	4.1	3.4	3.0	2.9	3.6	3.5	4.1	3.1	3.6
Salesforce.com	10	4.3	4.0	4.1	3.4	3.7	3.9	4.4	4.4	4.3
Other	51	3.3	3.7	2.8	2.5	3.1	3.3	3.7	3.1	3.2
Overall	227	4.1	4.0	3.7	3.5	3.9	4.0	4.4	3.7	4.0

*Satisfaction ratings have been calculated by assigning responses with values from "very good" = 5 to "very poor" = 1, and then calculating an average

Рисунок 2 — Анализ CRM-систем, проведенный в 2020 году

При составлении критериев для сравнения были использованы собранные требования от заказчика. Целью данного анализа является выявление сильных сторон конкурентов и внедрение новых требований в разарабатываемый проект. Жирным шрифтом в таблице 3 выделены ключевые критерии для сравнения.

[3] https://www.theaccessgroup.com/charity-crm/

[4] https://www.harlequinsoftware.co.uk/software-services/charity-crm/

[5] https://www.donorperfect.com/fundraising-software/features/

Таблица 3 — Детальный анализ конкурентов

	Donorfy	**Beacon**	**Thankq**	**Harlequin**	**DonorPerfect**
Обработка заявок	7	8	8	8	8
Ролевая модель	6	10	5	5	6
Аналитические панели	9	9	10	8	8
Интеграция с мобильным приложением	0	5	0	0	10
Администрирование	10	10	7	8	8
Интеграция с блокчейн	0	0	0	0	0
Категоризация заявок	5	7	10	8	10
Мониторинг пожертвований	10	10	10	8	8
Чат поддержки	0	0	0	8	8
Управление контентом фонда	0	0	0	8	8
Настраеваемые бизнес процессы	7	6	7	8	8
Просмотр логов системы	10	8	10	8	8
Надежность	10	10	10	8	8
Пуш уведомления	5	5	10	8	8
Поддержка нескольких языков	0	0	5	5	5
Финальная оценка	**5,26**	**7.1**	**5.9**	**4.7**	**6.9**

Одни из главных требований заказчика было создание заявки от имени фонда на сбор средств для Закята (см. раздел определений). Для этого необходим следующий функционал: создание заявок от имени фонда, а также категоризация заявок (и управление категориями).

Так как при анализе конкурентов упор делался на следующий функционал: обработка заявок, категоризация заявок, администрирование, пуш уведомление, мультиязычность и ролевая модель, то при подсчете финальной оценки был применен следующий алгоритм:

Критерии оценки были разбиты на следующие группы, так как для каждой категории был задан вес (от 0 до 10, где 10 – самое важное) по важности:

1. Функционал по обработке заявок – 10

2. Наличие функционала по администрированию пользователями платформы – 10
3. Наличие функционала по обработке заявок – 9
4. Категоризация заявок – 9
5. Ролевая модель – 9
6. Наличие пуш уведомлений – 8
7. Мониторинг пожертвований – 8
8. Аналитические панели – 8
9. Интеграция с системой блокчейн – 8
10. Интеграция с мобильным приложением – 8
11. Чат поддержки – 7
12. Настраеваемые бизнес-процессы – 9
13. Просмотр логов системы – 6
14. Надежность – 9
15. Поддержка нескольких языков – 10

Далее при подсчете финальной оценки применялся метод подсчета взвешенного среднего арифметического[6].

1.6 Преимущества ПО по сравнению с аналогами

Таким образом, можно сделать следующие выводы:

– Функциональсть администратора должна предполагать управление пользователями системы, а также должна быть возможность заблокировать пользователей системы и тем самым ограничить их доступ в систему;

– Необходима функциональность по управлению информацией о фонде (описание, документы, часто задаваемые вопросы) - такого не представленно ни в одной из перечисленных CRM систем, так как только одна из них интегрирована с мобильным приложением для пользователей;

– Уведомления сотрудников фонда о действиях в системе - очень важная часть, в большинстве существующих CRM-систем реализована через email-рассылки или интеграции со сторонними сервисами (настраеваемый функционал);

– Очень важны аналитические панели - корректная и правильная аналитика должна дать сотрудникам фонда представление о статистике собранных средств; В некоторых существующих решениях есть возможность настроить аналитические панели и демонстрировать только релевантную информацию;

[6] `https://en.wikipedia.org/wiki/Weighted_arithmetic_mean`

Выводы по главе

В рамках аналитической стадии разработки был проведен анализ устройства рынка, будущих пользователей. Также были выявлены существующие аналоги: они были разделены на косвенных и прямых конкурентов разрабатываемого Web-приложения. Глубокий анализ основных конкурентов позволил выявить слабые стороны Web-приложений в сфере программного обеспечения для управления деятельностью благотворительных фондов. Это позволило скорректировать требования, описанные в техническом задании (см. Приложение А).

Глава 2. Описание разработанных моделей и алгоритмов

2.1 Сценарии использования

На этапе проектирования была смоделирована диаграмма прецедентов по стандартам UML[6]. Перед составлением диаграммы были перечислены основные разделы приложения со следующим функционалом:

- Личный профиль – раздел по управлению личным профилем и настройками приложения (выбор языка, разрешение пуш-уведомлений);
- Заявки – полная обработка поступающих заявок на пожертвования (бизнес процесс описан в разделе 2.2), а также голосование членов комиссии по заявка (этот бизнес процесс подробно описан в пункте 2.4);
- Уведомления – раздел пуш-уведомлений, а также получения информации о смене статусов обрабатываемых заявок в системе блокчейн;
- Пожертвования – раздел с мониторингом пожертвований, поступающих от пользователя;
- Управление пользователями – мониторинг и управление пользователями системы, редактирование и регистрация в системе;
- Управление контентом фонда – управление информацией о фонде, разделом «Часто задаваемые вопросы» и «Новости»;
- Чат поддержки с пользователями – раздел диалогов с пользователями;

На диаграмме 3 отражены основные юзкейсы работы системы, а также приведены акторы, которые взаимодействуют с системой. В нижеприведенной таблице 4 представлены подробные описания каждого актора.

Таблица 4 — Акторы диаграммы прецедентов

Актор	**Описание**
General user	Обобщение для всех пользователей CRM.
Admin	**Администратор** управляет всеми учетными записями в системе. У него есть доступ к логам системы, а также возможность просматривать пользователей системы и регистрировать их.
Operator	**Оператор** отвечает на сообщения пользователей, отправленные через чат.
Content Manager	**Менеджер контента** системы имеет возможность редактировать описание фонда, а также раздел часто задаваемых вопросов.
Manager	**Менеджер фонда** имеет доступ к заявкам фонда, а также может обрабатывать данные заявки: менять статусы, редактировать и общаться с пользователем в рамках одной заявки. Также менеджер может просматривать транзакции фонда.

Продолжение на следующей странице

Таблица 4 – продолжение таблицы

Актор	Описание
SuperManager	**Комиссия фонда** имеет доступ к расширенному функционалу управления заявками, а именно к активации заявки (активная заявка становится видимой для всех пользователей системы, на нее можно жертвовать деньги).
Firebase	Система рассылки уведомлений.

Рисунок 3 — Диаграмма прецедентов

Также было составлено краткое описание каждого юзкейса в таблице 5.

Таблица 5 — Акторы диаграммы прецедентов

№	Название	Актор(ы)	Описание
Аутентификация			
1	Sign In	General User	Авторизация всех пользователей в приложении происходит с помощью ввода почты и пароля, указанных при регистрации в системе
Управление пользователями			
2	CRU of users	Admin	Создание, редактирование, просмотр пользователей системы. Администратор имеет возможность: зарегистрировать пользователя в системе, просмотреть список пользователей и информацию о каждом, изменить информацию: заблокировать или разблокировать пользователя;
3	See logs	Admin	Просмотр действий пользователей в системе: авторизация, действия по заявкам, совершенные пожертвования и т.д;
4	View fund employees	Supermanager	Член комиссии имеет возможность просматривать информацию о сотрудниках фонда, а также их заявки
Заявки			
5	CRUD application comments	Manager, Supermanager	Менеджер, комиссия и пользователь могут оставлять комментарии под заявками. Менеджер и комиссия могут это делать для всех заявок.
6	Approve application	Supermanager	Комиссия может одобрять заявки, таким образом они становятся активны и на них пользователи могут жертвовать деньги, после одобрения заявка передается в блокчейн.
7	Update and read application	Manager	Менеджер может просматривать заявки, а также переводить заявки в разные статусы, редактировать ее.
8	Create and activate application	Supermanager	Член комиссии имеет возможность создать заявку за другого человека, при создании заявки она автоматически появляется в реестре как активная
9	Vote fot the application	Supermanager	Член комиссии, назначенные категории которого совпадают с категорией заявки, имеет возможность проголосовать за или против активации заявки
Пожертвования			
10	Read donation	Supermanager	Комиссия фонда может просматривать все совершенные транзакции в рамках системы.

Продолжение на следующей странице

Таблица 5 – продолжение

№	Название	Актор(ы)	Описание
11	Distribute collected money	Supermanager	Комиссия фонда может направить собранные средства со счета фонда на одну из заявок.
12	Create donation	Supermanager	Комиссия фонда может создать запись о пожертвовании, совершенном вне системы.
Личный профиль			
13	Update and read profile	General User	Все пользователи системы могут просматривать информацию о себе, а также ее менять.
Категории			
14	CRUD categories	Supermanager	Суперменеджеры могут изменять категории, которые ассоциируются с заявками.
Информация о фонде, чаты			
15	CRU support chat	Operator	При возникших вопросах пользователь приложения может обратиться за помощью, написав в чат, где операторы ответят на все его интересующие вопросы.
16	CRU fund info	Content manager	Контент-менеджер может редактировать информацию о фонде
17	CRU faq	Content manager	Контент-менеджер может редактировать часто задаваемые вопросы о фонде
18	CRU news	Content manager	Контент-менеджер может редактировать новостную ленту
19	Get fund info, FAQ, news	General User	Пользователи системы могут получить информацию о фонде
Нотификации			
20	Notify about status change	Firebase	Рассылка уведомлений об изменении статуса заявки пользователю, создавшему заявку.
21	Receive notifications	General User	Пользователь может получать нотификации от системы.
22	Read blockchain status change	General User	Пользователь может просматривать информацию о смене статусов операций в системе.

2.2 Модель предметной области

На этапе анализа была разработана диаграмма предметной области 4 разрабатываемого приложения по стандартам UML[6]. Как упоминалось ранее в Главе 1, разрабатываемое приложение направлено на интересы сотрудников фонда и самого фонда.

Так как модель предметной области представляет собой совокупность понятий предметной области и отношений между ними (т.е. совокупность поведения и данных), то при ее разработке я в первую очередь руководствовалась use-case диаграммой 3, а также бизнес-правилами из документа «Техническое задание. CRM-система для благотворительного фонда «AIAIN». Web-приложение для сотрудников фонда » (см. Приложение А). В целом главной чертой, отличающей одну модель предметной области от другой, является именно набор бизнес-правил.

В следующем разделе детально рассмотрен бизнес-процесс обработки заявки (см. диаграмму 7), но стоит упомянуть, что все нюансы процесса также были отражены на диаграмме предметной области: для каждой роли был введен свой класс (например `Manager`, `Operator`) и разграничены возможности пользователей взаимодействия с заявками, чатами, новостным контентов и так далее.

В процессе разработки модель предметной области дополнялась в соответствии с развивающейся системой и изменяющимися требованиями заказчика.

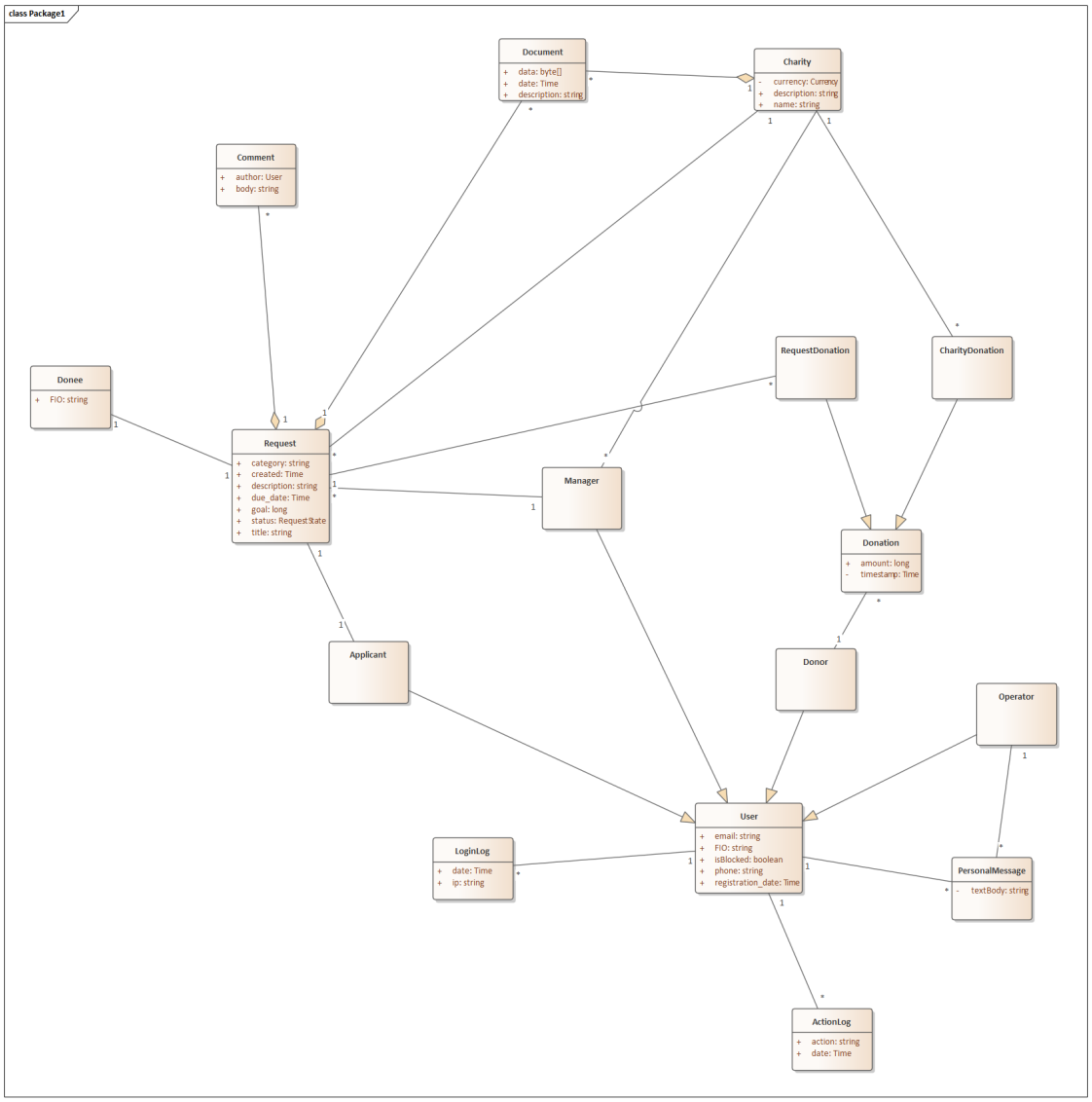

Рисунок 4 — Диаграмма предметной области

2.3 Модель обработки заявки

При моделировании бизнес процесса обработки заявки от пользователя на сбор средств была составлена диаграмма Business Process Model and Notation (BPMN [20]), в которой был смоделирован бизнес-процесс и показано, как модули CRM-системы взаимодействуют между собой (см. диаграмму 5).

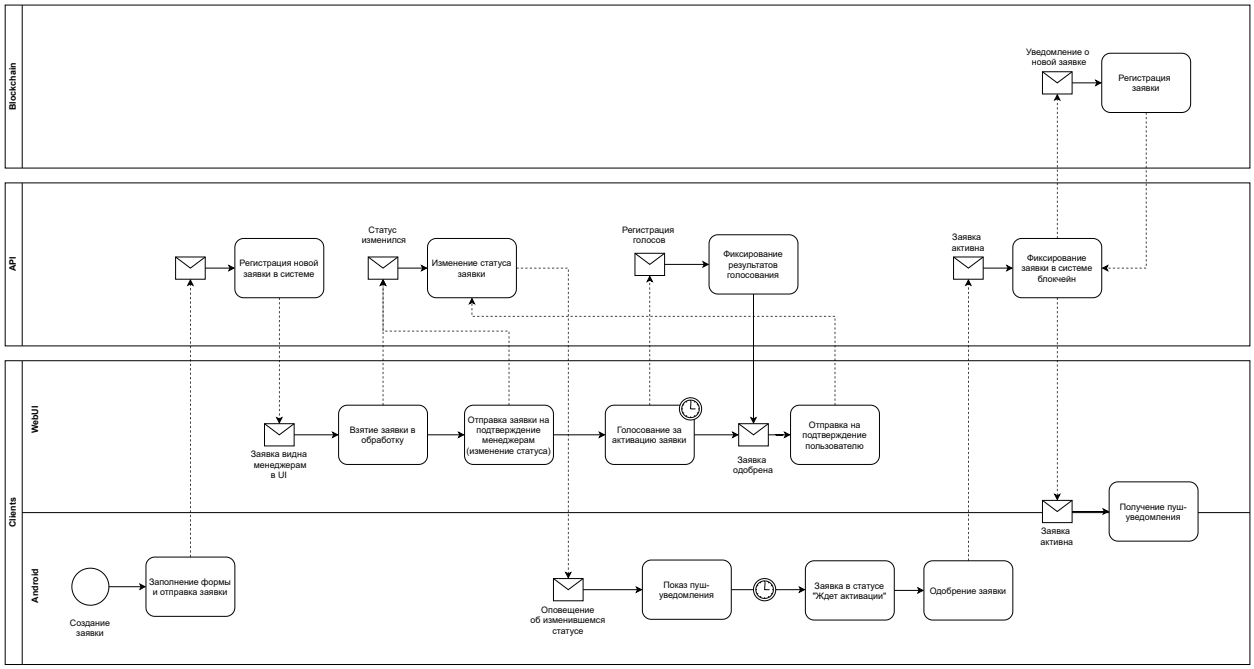

Рисунок 5 — BPMN обработки заявки всей системы

Подробная диаграмма взаимодейтсвия данного бизнес процесса представлена в разделе 3.7.1 главы 3. Рассмотрим подробнее, что происходит на диаграмме 5 бизнес-процесса обработки заявок. Пользователь мобильного приложения (нуждающийся) создает заявку в мобильном приложении, созданная заявка в статусе «Новая» (подробнее модель статусов рассмотрена ниже в разделе, в таблице 7) заносится в базу данных через API системы. В Web-интерфейсе сотрудника фонда появляется только что созданная заявка, которую сотрудник с ролью «Член комиссии» или «Менеджер» (все роли пользователей представлены в разделе 1.4.3) берет в обработку. Статус заявки меняется на «В обработке» и изменения также заносятся в базу данных через API системы. При каждом изменении статуса заявки мобильный пользователь получает пуш-уведомления. Далее происходит голосование по заявке, бизнес процесс которого описан в разделе 2.4, а полная диаграмма последовательности в разделе 3.7.2. После принятия положительного решения сотрудник фонда отправляет заявку на финальное подтверждение пользователю, все изменения заносятся в базу данных. Пользователь мобильного приложения одобряет заявку, данные изменения фиксируются в базе данных, заявка заносится в систему блокчейн. Оба пользователя (сотрудник фонда и нуждающийся) получают пуш-уведомления об успешной активации заявки.

Также была проработана более упрощенная модель обработки заявки непосредственно для Web-приложения, чтобы выделить непосредственные шаги, которые должны быть проведены членом комиссии для активации заявки, см. диаграмму 6.

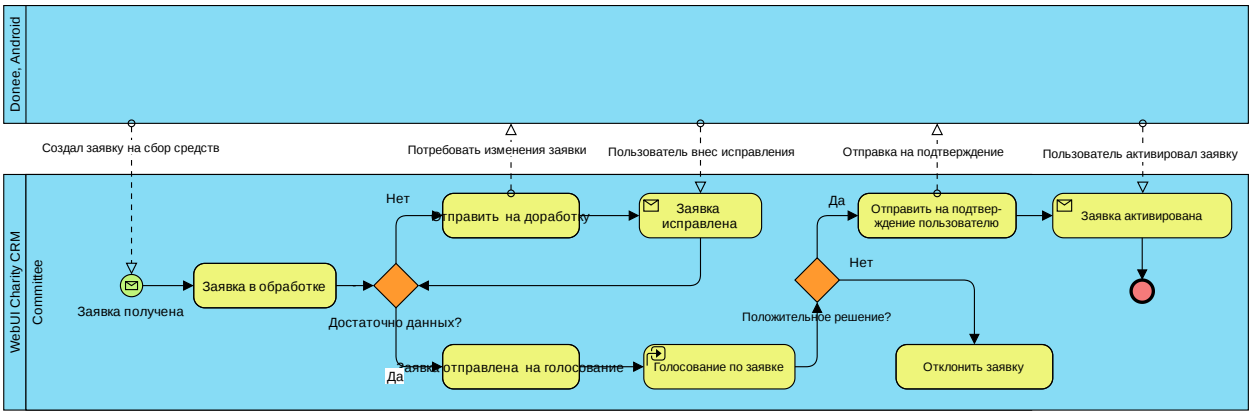

Рисунок 6 — Диаграмма BPMN, упрощенная версия для Web-приложения

На этапе анализа была проработана модель обработки заявки на сбор средств. Так как обработка заявки на сбор средств является главной функциональностью пользователей с ролями «Менеджер», «Член комиссии», то была составлена диаграмма 7 состояний для сущности «заявка на пожертвование».

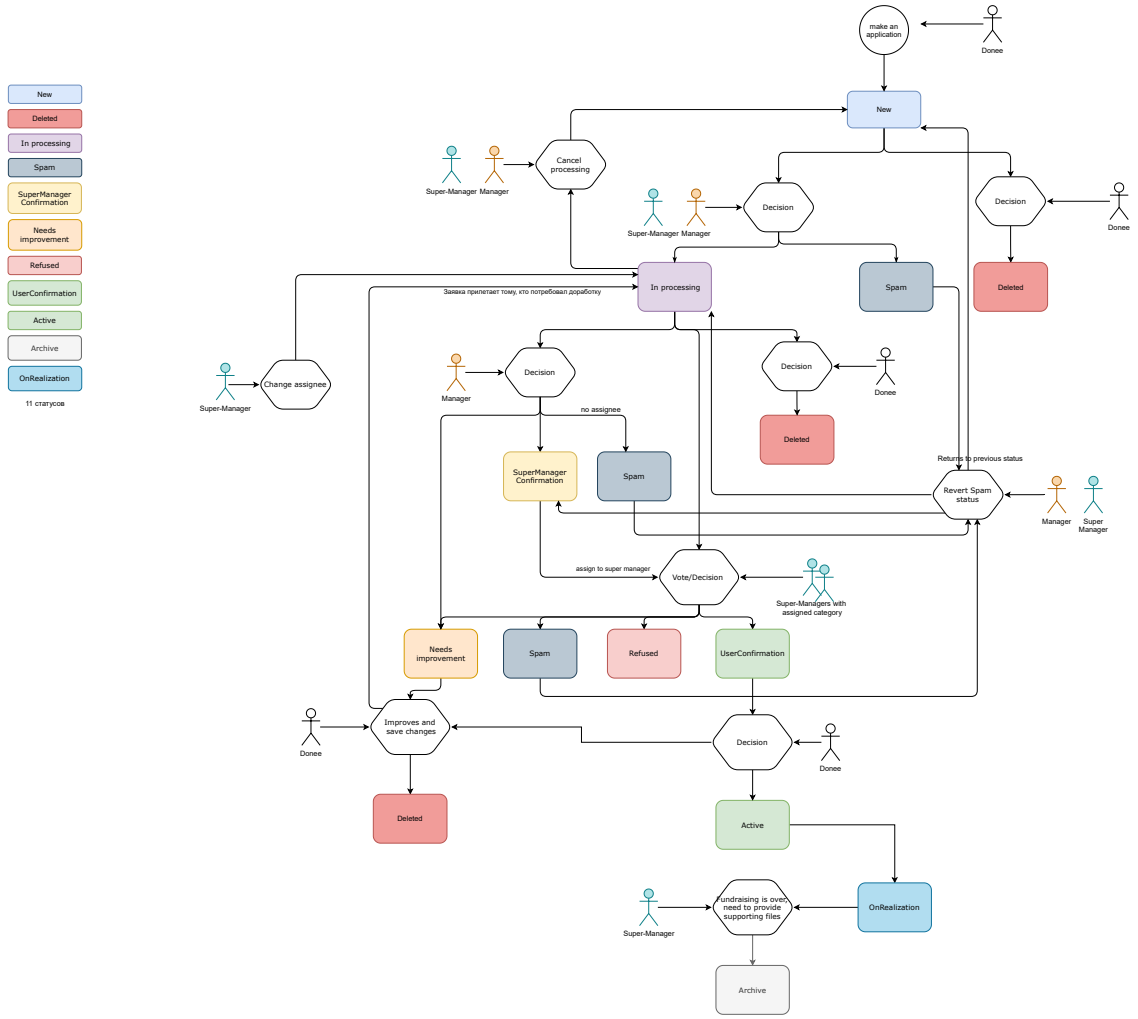

Рисунок 7 — Диаграмма статусов

При сборе требований заказчик сделал упор на бизнес-процесс обработки заявки, а именно: ограничение прав на совершение определенных переходов заявки (прим. обыч-

ный менеджер не может активировать заявку), а также на ограничение доступных статусов для перехода на каждом этапе заявки. Таблица 6 отражает разграничение действий по ролям и органичения на переходы статусов. Описание статусов представлено в таблице 7.

Таблица 6 — Переходы заявки в различные статусы

Кто	**Действие**	**Статус**	
		Предыдущий	*Следующий*
Менеджер Член комиссии	Взятие заявки в обработку	Новая	В обработке
Член комиссии	Отклонение заявки	В обработке	Отклонена
		Ждет активации	
Менеджер	Отправление заявки на доработку	В обработке	На доработке
Член комиссии		Ждет активации	На доработке
		В обработке	
Менеджер	Отправление заявки на активацию / отклонение	В обработке	Ждет активации
Член комиссии	Активация заявки	Ждет активации	Ждет подтверждения
		В обработке	
Менеджер	Пометка заявки как спам	Новая	Спам
		В обработке	
Член комиссии		Ждет активации	
Член комиссии	Закончился сбор средств (либо автоматически закрывается сбор)	Активная	Архив
Член комиссии	Вернуть заявку статуса "спам"в предыдущий	Спам	Новая
Менеджер			В обработке
			Ждет активации
Менеджер Член комиссии	Отказаться от обработки	В обработке	Новая
Менеджер	Отменить доработку	На доработке	В обработке
Член комиссии			Ждет активации
Член комиссии	Изменение assignee у заявки	В обработке	В обработке

Таблица 7 — Статусы заявки

№	**Название статуса на английском**	**Название статуса на русском**	**Описание**
1	New	Новая	Только что созданная заявка от пользователя
2	In progress	В обработке	Заявка находится в обработке у одного из менеджеров. Пока заявка в этом статусе, менеджер просматривает документы и удостоверивается в подлинности заявки.
3	Spam	Спам	Заявка помечена как спам.
4	Needs Improvement	На доработке	Не указано достаточно данных для подтверждения заявки, менеджеру требуются дополнительные сведения, он может оставить комментарий - причину отказа. Есть возможность отредактировать.
5	Archieved	Архив	Заявка либо устарела, либо деньги были собраны.
6	Rejected	Отказано	Отказано в рассмотрении заявки;
7	Active	Активная	Заявка прошла все стадии подтверждения и открыта к сбору средств;
8	Supermanager Confirmation	На подтверждении у комиссии	Прежде чем быть опубликованной, заявка должна быть одобрена членом комиссии;
9	User Confirmation	На подтверждении у пользователя	Прежде чем быть опубликованной, финальная версия заявки должна быть одобрена пользователем;
10	Deleted	Удалена	Пользоатель имеет право удалить свою заявку, таким образом она больше не будет рассматриваться менеджерами и претендовать на публикацию;
11	OnRealization	В реализации	Фонд перечислил средства нуждающемуся, но деньги еще не пошли на благое дело (не были получены документы, подтверждающие реализацию средств);

2.4 Модель голосования по заявке

При моделировании бизнес процесса голосования по заявке членами комисии, чьи назначенные категории совпадают с категорией заявки, была составлена диаграмма Business Process Model and Notation (BPMN [20]), в которой был смоделирован бизнес-

процесс и показано, как модули CRM-системы взаимодействуют между собой (см. диаграмму 10).

Прежде чем рассматривать подробнее бизнес-процесс голосования по заявке стоит отметить, что у каждого члена комиссии при регистрации или при редактировании профиля администратором, могут быть назначены категории. Категории заявок системы доступны к просмотру и редактированию каждому члену комиссии фонда: в отдельном разделе категорий (см. скриншот 8). Каждая категория состоит из уникального ID, а также перевода названия категории на английский, русский и арабский.

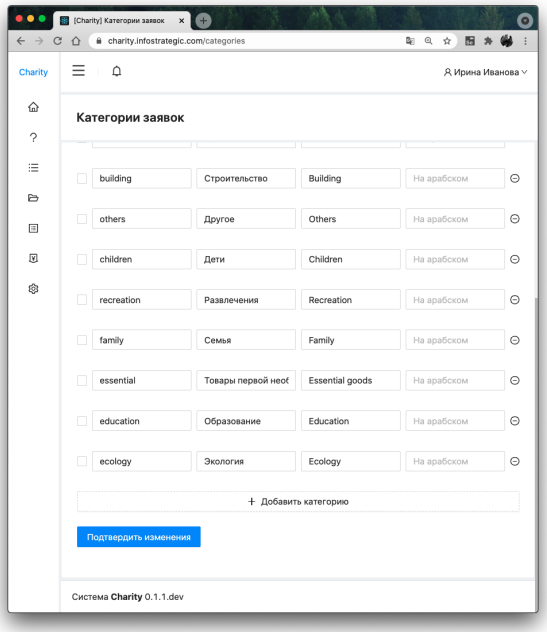

Рисунок 8 — Скриншот просмотра и редактирования категорий

Рассмотрим подробнее происходящее на диаграмме: сотрудник фонда с ролью «Член комиссии» инициирует переход на карточку заявки в статусе «Ждет подтверждения от члена комисии». Пример UI представлен на скриншотах 9: в зависимости от того, соответствует ли назначенная категория члена комисии категории заявки, сотрудник фонда видит либо экран с результатами голосования (слева), либо с результатами и возможностью проголосовать самому (справа).

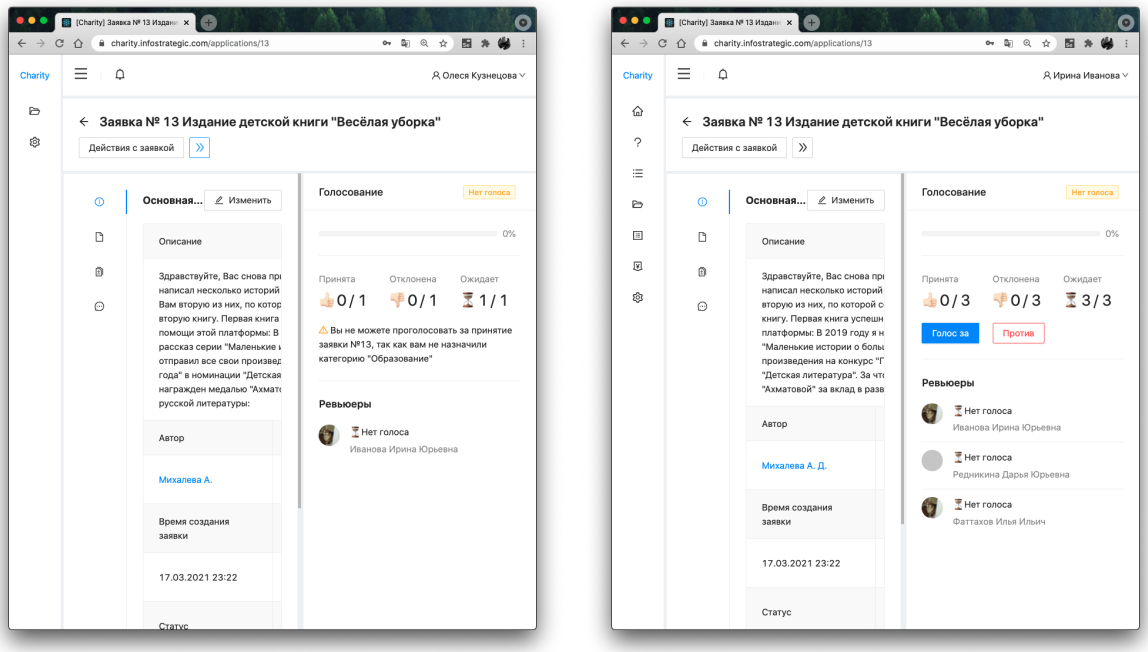

Рисунок 9 — Скриншоты голосования за активацию заявки, член комисии

Далее член комиссии, который может проголосовать по заявке, совершает свой выбор. Данный выбор фиксируется в API и обновленные данные по голосованию демонстрируются на WebUI. Далее, если все ревьюеры проголосовали и проголосовали положительно, то у члена комиссии появляется возможность изменить статус заявки на «Ждет активации пользователя», тем самым продвинуть ее дальше на активацию. Если же по завершению голосования результаты расходятся, то у члена комисии появляется возможность отклонить заявку, переведя ее в статус «Отклонена». Если же не все ревьюеры успели проголосовать по заявке, то нужно дожидаться пока все проголосуют и дальше будет возможно совершить вышеперечисленные действия. Итогом бизнес-процесса является изменившийся статус заявки.

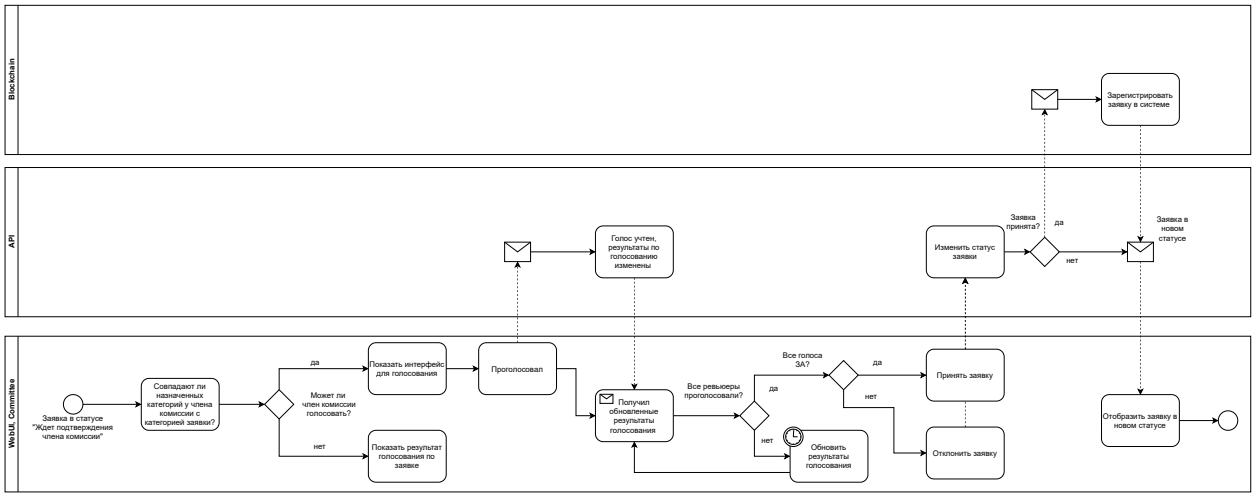

Рисунок 10 — BPMN голосования по заявке

2.5 Процесс получения собранных средств

Для фонда «AIAIN» важным бизнес-процессом является не только сбор средств, но и их перевод. Важно также не только перевести собранные средства на счет нуждающимся, но также и убедиться в законности распределения собранных средств (см. диаграмму 11).

При моделировании данного бизнес процесса была составлена диаграмма Business Process Model and Notation (BPMN [20]). Главные участники бизнес процесса: член комиссии (использует Web-интерфейс) и нуждающийся (использует Android-приложение). Второе действующее лицо процесса – банк, через который происходит передача собранных средств нуждающимся. При реализации данного приложения была опущена реализация интеграции с банковской системы, но для будущего развития проекта были сделаны заглушки для дальнейшей интеграции.

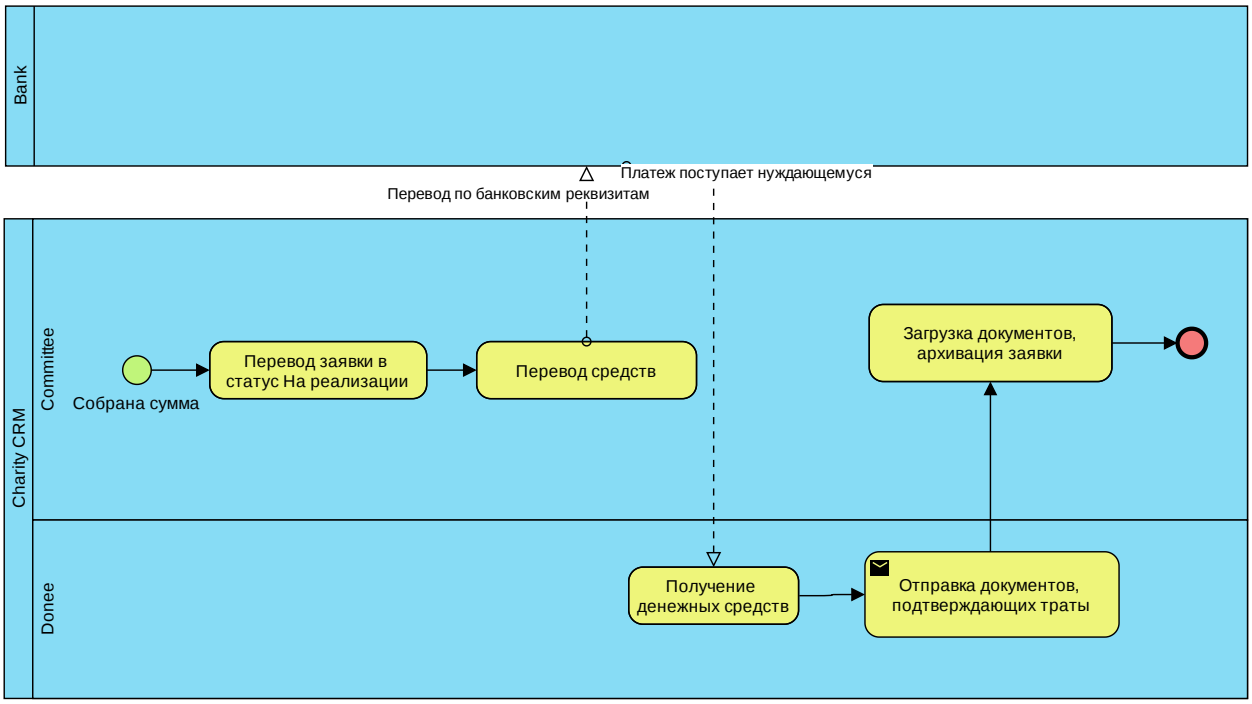

Рисунок 11 — BPMN перевода средств нуждающемуся

При детальном рассмотрении диаграммы можно увидеть, что бизнес-процесс начинается, когда сумма на заявку собрана (или закончился срок заявки). Член комиссии с помощью Web-приложения переводит заявку в статус «На реализации». Далее собранные средства со счета фонда по банковским реквизитам переводятся нуждающемуся. О потраченных средствах нуждающийся спустя время должен предоставить документы, прислав письмо на почту фонда (или прислав их в чат поддержки). После получения документов уже член комиссии загружает подтверждающие документы и тем самым архивирует заявку, сообщая о ее успешном завершении.

2.6 Процесс выплаты закята

При моделировании бизнес процесса сбора средств на выплату закята была составлена диаграмма Business Process Model and Notation (BPMN [20]), в которой был смоделирован бизнес-процесс и показано, как модули CRM-системы взаимодействуют между собой (см. диаграмму 12).

Прежде всего стоит отметить, что закят - ежегодный налог в Исламе, собранные средства с которого идут на помощь нуждающимся единоверцам. Одно из главных требований арабского фонда «AIAIN» состояло в том, чтобы автоматизировать сбор самого закята и распределение собранных средств между нуждающимимся.

Первым этапом бизнес-процесса является создание приватной категории «Закят», которая доступна к выбору непосредственно во время создания заявки членом комиссии от имени фонда. Приватные категории, созданные членами комиссии, недоступны для выбора пользователям при создании заявок на сбор пожертвований.

После создания приватной категории «Закят», член комисии инициирует создание заявки от имени фонда. Заявка, созданная с категорией «Закят» может быть создана только по инициативе фонда, таким образом она сразу становится активной, минуя валидацию и подтверждения от членов комисиий. Срок сбора по заявке с категорией «Закят» заканчивается, когда наступает Рамадан (определение можно найти в разделе «Понятия и определения»). Собранные средства по заявке с категорией «Закят» распределяются через форму распределения средств между другими заявками фонда.

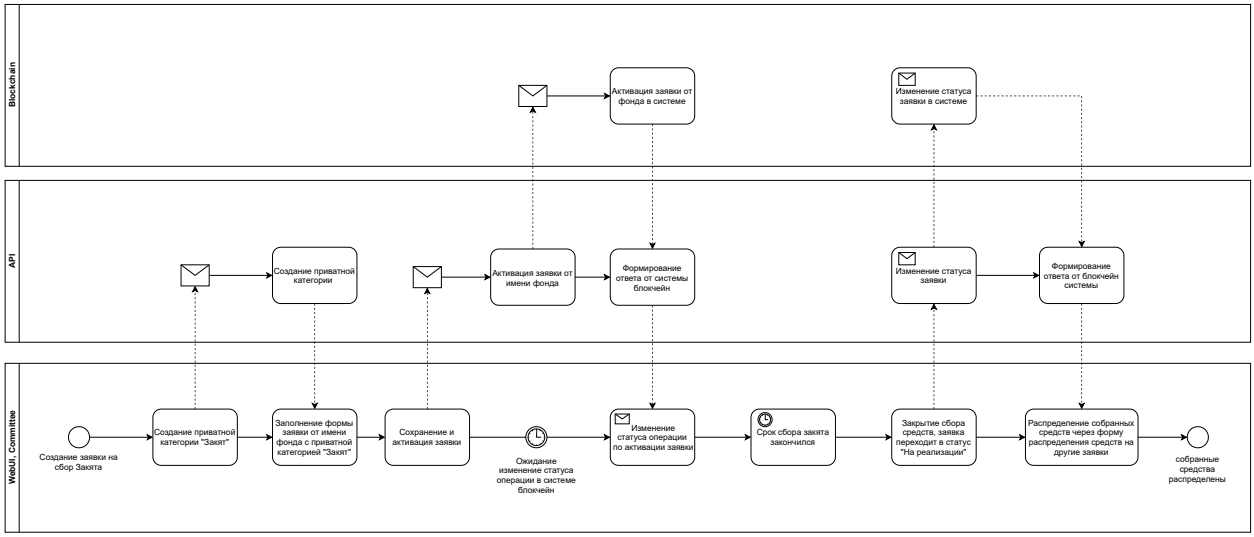

Рисунок 12 — BPMN процесс сбора средств и выплаты закята

2.7 Ролевая модель

Одной из ключевых возможностей приложения является разделение функциональности Web-приложения: только пользователи с определенной ролью могут использовать некоторые функции приложения, а также возможности пользователей с разными ролями могут пересекаться (прим. просмотр заявок у членов комиссии и менеджеров).

При выборе подхода к проектированию ролевой модели в приложении был выбрана модель `Role-Based Access Control` [14]. Идея данного похода заключается в хранении прав на совершение действий в зависимости от роли пользователя. При использовании `RBAC` нужно проанализировать потребности будущих пользователей и сгруппировать их по ролям на основе общих обязанностей. Далее нужно назначить одну или несколько ролей пользователям и соответствующие разрешения для каждой роли. Таким образом не нужно каждый раз рассматривать каждого пользователя отдельно, вместо этого достаточно применить подход разделения «роль – привелегия».

Преимущества подхода

Преимущества подхода RBAC следующие:

- Создание систематического, легко адаптируемого решения с назначением разрешенных действий;
- Можно легко добавлять, а также проверять права пользователей;
- Легкое и быстрое изменение самих ролей;
- Подход дает возможность более эффективно соблюдать нормативные и законодательные требования в отношении конфиденциальности и неприкосновенности частной жизни;

Роли

Роли – это набор разрешений, применяемых к пользователям. Использование ролей упрощает настройку разрешений, специальных функций, доступных узкому кругу лиц. По мере увеличения масштабов и сложности пользовательской базы роли становятся особенно полезными. Так как требования заказчика могут корректироваться, а следовательно и расширяться возможности пользователей, а также появляться новые роли, то этот подход прекрасно позволяет адаптироваться к таким изменениям.

Адаптация подхода к приложению

Что касается разрабатываемого приложения, то как было описано в разделе 1.4.3, у каждого пользователя должна быть своя роль и в зависимости от нее - набор функций, доступных для выболнения.

При разработке аналитической модели приложения была создана неформальная модель будущих пользователей с перечислением функций для каждой из роли в формате словаря.

Выводы по главе

На данном этапе разработки ПО были получены следующие артефакты: модель прецедентов (см. диаграмму 3) и модель предметной области (см. диаграмму 4). Также был детально проработан бизнес-процесс обработки заявки различными пользователями системы (см. диаграмму 7).

Глава 3. Проектирование Web-приложения

3.1 Выбор технологий

Разработка Web-приложения началась в ноябре 2020 в рамках предмета «Командный проект» ОП «Программная инженерия». Только в марте 2021 года после успешной защиты проекта было принято решение продолжить разработку Web-клиента в качестве дипломной работы. Так как в начале разработки предполагалось, что будущее развитие клиентского приложение будет продолжено членами команды, то выбор пал в сторону стека: *React*, *Typescript*. При продолжении разработки было принято решение использовать язык `Elm` для написания отдельных компонентов приложения.

3.1.1 Выбор языка разработки

При выборе языков разработки для сравнения использовались наиболее популярные языки для Web-разработки в 2021 году [13].

Сравнение языков для разработки

JavaScript - это высокоуровневый, динамический, нетипизированный интерпретируемый язык программирования. За последние несколько лет JavaScript сохранил свои лидирующие позиции в области разработки корпоративных приложений [10]. В таблице 8 сравнения языков для разработки Web-приложений можно выделить главный недостаток Javascript - большое количество ошибок at runtime, отсутствует типизация;

Typescript – это язык с открытым исходным кодом, который основан на JavaScript, одном из наиболее часто используемых в мире инструментов, путем добавления определений статических типов. Из плюсов (см. Таблицу 8) стоит выделить большую поддержку языка - все больше и больше приложений с Javascript переписываются на Typescript из-за схожести языков. Но из минусов все те же исключения at runtime, нестрогая типизация. Из преимуществ typescript - он компилируется в Javascript и можно использовать все те же популярные библиотеки, написанные на JavaScript в проекте;

Purescript – это чисто функциональный и строго типизированный язык программирования, созданный Филом Фриманом. Он нацелен на обеспечение сильной совместимости с доступными библиотеками JavaScript, по духу схожими с Haskell, но с сохранением JavaScript в своей основе. Сильной стороной PureScript является его минимализм. В нем нет библиотек для функций, которые считались бы необходимыми для других языков. Например, вместо включения генераторов и обещаний в сам компилятор вы можете использовать определенные библиотеки для задачи. Вы можете выбрать желаемую реализацию для нужной вам функции, которая обеспечивает высокоэффективную и персонализированную работу при использовании PureScript, сохраняя при этом сгенерированный код как можно меньше.

Elm – это чисто функциональный язык программирования, который может компилироваться в JavaScript, HTML и CSS. Вы можете создать полноценный сайт, используя только Elm, что делает его отличной альтернативой фреймворкам JavaScript, таким как React. Приложения, которые создаются с его помощью, автоматически используют виртуальную библиотеку DOM, что делает его очень быстрым. Один большой плюс - это встроенная архитектура, с которой можно забыть о потоке данных и вместо этого сосредоточиться на объявлении данных и логике.

CoffeeScript – это язык, который нацелен на раскрытие хороших частей JavaScript, обеспечивая при этом более чистый синтаксис и сохраняя семантику на месте. Хотя

в последние годы популярность языка снижается, он меняет направление и недавно получил новую основную версию, обеспечивающую поддержку функций ES2015 +. Код, который написан на CoffeeScript, напрямую переводится в читаемый код JavaScript и поддерживает совместимость с существующими библиотеками.

Scala – этот язык объединяет объектно-ориентированное и функциональное программирование на одном кратком языке высокого уровня. Статические типы Scala помогают избежать ошибок в сложных приложениях, а его среда выполнения JVM и JavaScript позволяет создавать высокопроизводительные системы с легким доступом к огромным экосистемам библиотек.

Таблица 8 — Сравнение языков программирования для разработки Web-приложений

Язык	**Плюсы**	**Минусы**
Javascript	Большое коммьюнити; Простота в изучении и написании кода; Совместимость с другими языками; Популярность срези enterprise разработки; Большое количество библиотек и их поддержка сообществом;	Исключения в процессе исполнения; Нет проверки типов; Нетипизируемый; Недостаток тулинга для дебага приложения;
Typescript	Аналогичные плюсы JavaScript, а также наличие типов - писать код становится легче и все реже появляются ошибки во время исполнения;	Проверка типов есть, но очень базовая; Недсотаток тулинга для дебага приложения;
Purescript	Функциональный язык программирования; Строгое типизирование; Нет ошибок во время исполнения;	Высокий порог вхождения в язык; Маленькое коммьюнити и библиотек намного меньше и плохо поддерживаются;
Elm	Функциональный язык программирования; Встроенная архитектура model view update, не надо использовать сторонние фреймворки; Никаких исключений во время исполнения; Прекрасная производетельность по сравнению с остальными;	Высокий порог вхождения; По сравнению с js маленькое коммьюнити, а также набор библиотек для UI;
Scala	Совместимость с Java – можно использовать библиотеки, написанные как на Scala, так и на Java; Функциональная и ОО парадигмы; Строгая типизация;	Фреймворки для UI не такие разнообразные, как для Javascript и Typescript;

Обоснование выбора

Я остановилась на использовании **Typescript** и **Elm** для создания Web-приложения. Так как один из плюсов языка Typescript в том, что у языка большая база библиотек, а

также можно использовать библиотеки, написанные на языке JavaScript, то было легко найти подходящую библиотеку с готовыми компонентами для реализации UI, а также библиотки для кодогенерации запросов API, реализации переводов и так далее. Более подробно выбор библиотек описан в разделе 3.1.3.

Язык Elm привлек своей функциональной парадигмой, встроенной архитектурой и строгой типизацией. Также компоненты на Elm легко встраиваются [17] в код, написанный на Typescript и React, поэтому было легко адаптировать компоненты в уже существующий проект.

3.1.2 Выбор фреймворков

Так как на момент выбора фреймворка для разработки Web-приложения уже был выбран язык *Typescript* (а в последующем и *Elm*), то список был ограничен фреймворками, которые совместимы с этими языками: React, Angular, Vue, Dojo и другие [18]. Для того, чтобы уменьшить количество потенциальных решений, было принято решение сузить список исходя из метрики распространенности и популярности технологии. Так, в статье-сравнении Web-фреймворков от 2020 года [19] приведена статистика популярности github-репозиториев среди пользователей сообщества. Таким образом полное сравнение следующих Web-фреймворков было проведено: React/Redux, Angular, Vue, Elm.

Для сравнения выбранных фреймворков были выбраны использованы следующие метрики: производительность, размер (самого приложения), количество строк кода, а также развитость сообщества и количество библиотек. Результаты и анализ сравнения производительности, размера и количества строк кода были изучены по статье-сравнению 2020 года [19]. Результаты сравнения можно увидеть на рисунках 13 и 14. Со стороны метрики «Размер» наиболее лучшими вариантами являются приложения, написанные с помощью стека Elm и React, в то время как со стороны метрики «Производительность» фаворитом является Elm.

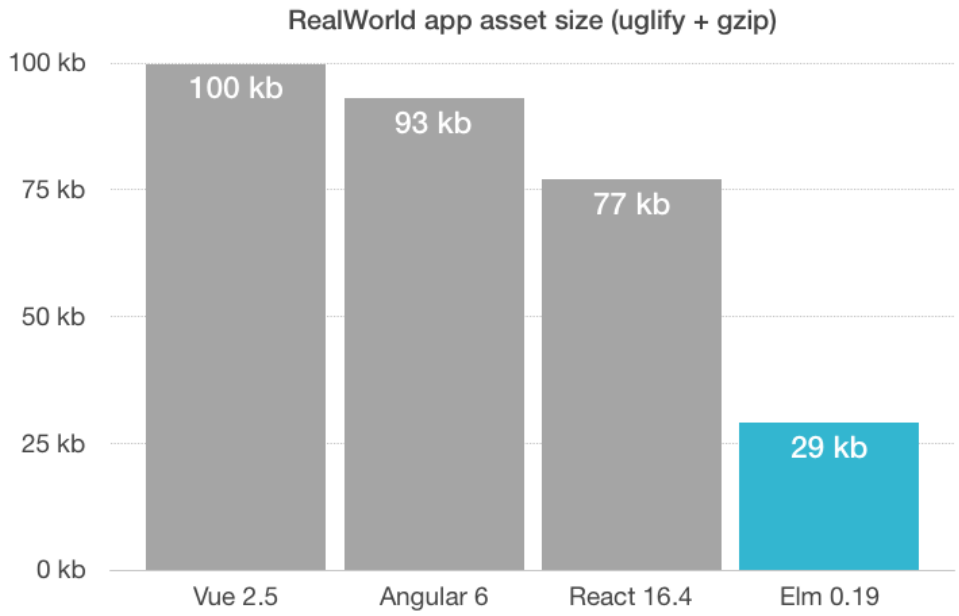

Рисунок 13 — Сравнение размера app assets [16]

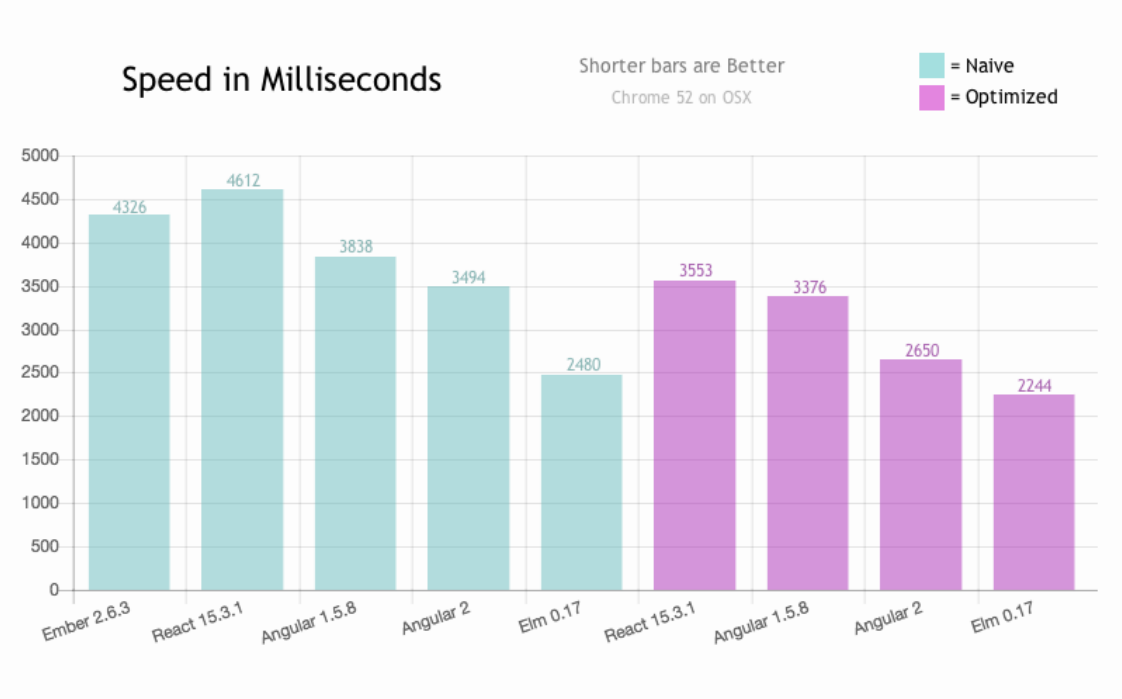

Рисунок 14 — Сравнение скорости в браузере Google Chrome [15]

Также стоит отметить, что стек React/Redux является на данный момент самым популярным среди профессиональных разработчиков [21], соответственно и коммьюнити вокруг стека развито очень хорошо. Об этом свидетельствует большое количество UI-фреймворков для написания React-компонент.

Обоснование выбора

Для написания приложения на языке *Elm* не нужен сторонний фреймворк, поэтому для написания основной части на языке *Typescript* был выбран фреймворк *React*, как наиболее подходящий под цели разработки. Большое профессиональное коммьюнити и широкий выбор библиотек позволят использовать все удобства и возможности Web-разработки, а хорошие показатели в метриках «Размер» и «Производительность» прекрасно характеризуют выбранный стек.

3.1.3 Выбор библиотек

Выбор библиотек для приложения состоял из двух этапов: изначально для части приложения, написанного на стеке React, Typescript, Redux были выбраны зависимости; позже при написании части кода на Elm были выбраны зависимости для этого стека, а также были подобраны зависимости для связывания частей, написанных на Elm и React, вместе.

React

Так как разрабатываемое приложение является клиент-серверным, то в первую же очередь были выбраны библиотеки, позволяющие взаимодействовать с API. Для отправки HTTP запросов выбор происходил между двумя вариантами: встроенным API для создания HTTP-запросов *fetch* и *axios*. Библиотеки отличаются деталями реализации, а также API, но в целом функции выполняют одни и те же. Ключевым моментом в выборе библиотеки для совершения API запросов была возможность кодогенерации

кода запросов из swagger схемы (и дальнейшая совместимость сгенерированного когда с библиотекой). Таким образом выбор пал на библиотеку **Axios** [22]. Ниже будем рассматривать вопрос с кодогенерацией кода.

Для кодогенерации кода запросов выбор библиотеки состоял из двух вариантов: *OpenAPI Generator* и *Swagger Codegen*. При сравнении обеих библиотек в первую очередь учитывались языки и фреймворки, с которыми они совместимы: обе поддерживают связку *Typescript* и ранее упомянутый *Axios*. Если рассматривать кодогенераторы более подробно, то можно увидеть, что Swagger Codegen является продуктом SmartBear, в то время как OpenAPI generator - это продукт, которым управляет само сообщество программистов. Более того, активность в репозитории OpenAPI generator выше, чем в Swagger Codegen. Таким образом выбор все-таки был в пользу **OpenAPI generator** [24]. Для проекта был написан yarn скрипт, который по находящейся в корне проекта Swagger спецификации в формате json генерирует файлы с методами/моделями для совершения запросов по описанному API.

При выборе библиотеки для дизайна компонент были важны следующие аспекты: поддержка коммьюнити, эстетичность, применимость компонент под цели и задачи проекта. Так как CRM-система предполагает панели для администрирования, поддержку большого количества реестров, фильтрации внутри реестров, расширенного поиска информации – это накладывает определенные ограничения на существующие библиотеки компонент. При выборе библиотеки были рассмотрены следующие варианты: *Ant.design*, *MaterialUI*, *SemanticUI*. Последние два варианта по набору компонент подходят больше для создания коммерческих сайтов, площадок для тороговли, в отличие от первого варианта. **Ant.design** был выбран, так как API позволяет настраивать компоненты под себя и прописывать собственную логику в поведение, к тому же UI подходит под цели и здачи проекта. По такому же принципу были выбраны библиотеки для визуализации аналитических дашбордов и диаграмм **Ant Design Charts**.

Одним из важнейших решений в выборе библиотек для рабзработки была библиотека для интернализации приложения. В одном из требований заказчика (см. Приложение А) в Web-приложени должна быть поддержка двух языков: русский и английский. При реализации механизма интернализации была выбрана библиотека **i18n** как самая популярная среди разработчиков React приложений. Библиотека поддерживает переводы множественных значений, а также имеет удобное и настраиваемое API.

Также было принято решение использовать встроенную библиотеку **WebSocket** для использования в чатах и диалогах для мнгновенного получения входящих сообщений. При выборе была также рассмотрена библиотека *Socket.IO*, которая является надстройкой над *WebSocket*, которая к тому же поддерживает функционал по поллингу и автоматическому реконнекту, если вдруг соединение прервется. Так как базовый функционал встроенной библиотеки *WebSocket* удовлетворяет потребностям проекта, то выбор был остановлен на ней.

Для получения пуш-уведомлений в проекте была использована библиотека **Firebase**, так как именно с помощью этой технологии бэкенд отправляет пуш-уведомления как на Web-клиент, так и на Android-клиент.

Elm

При выборе библиотек для разработки части приложения, написанной на стеке *Elm* помимо стандартных библиотек для http запросов были использованы следующие библиотеки:

– elm-explorations/markdown - для отображения markdown[11] в компоненте;

– mdgriffith/elm-ui - для динамической работы с css внутри Elm кода;

– webbhuset/elm-json-decode - для синтаксического сахара;

Для совместимости кода, написанного на Elm с кодом Typescript также использовалась библиотека **react-elm-components**, так как она поддерживает функциональность флагов и портов [17] в отличие от остальных существующих решений.

3.2 Архитектура приложения

В начале разработки клиентского Web-приложения была выбрана архитектура Single Page Application, далее SPA [27].

Главное преимущество выбора данной архитектуры - потрясающий пользовательский интерфейс, поскольку пользователи не ждут перезагрузки веб-страниц. Таким образом это позволяет пользователям использовать приложения, не дожидаясь загрузки страницы с сервера, что позволяет выиграть в производительности и более динамическом пользовательском опыте. Однако существуют и минусы у данной архитектуры - требуется больше усилий для поддержания состояния данных на странице, реализации навигации и значительного мониторинга производительности.

При выборе SPA архитектуры большое внимание было уделено выбору роутера (библиотеки для навигации между страницами Web-приложения). При детальном рассмотрении существующих решений было выделено два подхода к реализации межстраничной навигации:

– Большинство маршрутизатором для SPA используют синхронный роутинг для подгрузки страницы. Это означает, что изменение местоположение происходит до сопоставления роута. Порядок действий примерно такой: пользователь нажимает на ссылку, изменяется ссылка в url браузера, роутер находит подходящий переход и происходит новый рендеринг страницы;

– Асинхронный роутинг подразумевает что url не меняется до самого последнего момента: пока другие асинхронные действия не будут завершены (например загрузка данных). Порядок действий примерно такой: пользователь нажимает на ссылку, роутер определяет по какому url будет переход, если у данного перехода есть асинхронные действия, то они отрабатывают, только потом происходит рендер страницы;

Был выбран асинхронный роутинг для данного приложения, так как практически на каждой странице происходит асинхронная загрузка данных. Для реализации данного подхода был выбран CuriRouter [23] как самое лучшее решение для имплементации асинхронного роутера в React приложениях.

3.3 Клиент-серверное взаимодействие

При разработке Web-клиента для CRM-системы для благотворительного фонда «AIAIN» большое внимание уделялось клиент-серверному взаимодействию. На раннем этапе разработки системы было принято решение использовать архитектурный стиль REST [25] для взаимодействия компонентов между собой. В отличие от рассматриваемых вариантов (прим. GraphQL) REST является самым распространенным архитектурным стилем,

а также прекрасно подходит для клиент-серверных приложений. Еще одно удобство использования данного подхода заключается в том, что для REST-запросов легко писать документацию.

Также в начале разработки было принято решение писать Swagger документацию к endpoint-ам API [12], так как она хорошо читаема и с помощью таких сервисов как SwaggerHub [26] ее можно визуализировать. Также, как было упомянуто ранее в разделе 3.1.3, при проектировании Web-клиента был использован генератор кода API запросов по Swagger документации OpenapiGenerator [24] (см. диаграммы 16).

При выполнении команды по кодогенерации в папке проекта `@generated` образовывалась следующая структура, в которой содержались пакет `api` с файлами запросов, сгруппированных по тематике, а также пакет `model` с многочисленными моделями, использующимися в самих запросах (см. скриншоты структуры проекта 15).

Для иллюстрации взаимодействия API и Web-клиента приведем пример JSON ответа на запрос получения GET запроса новости фонда (см. листинг 1).

```
{
  created_at: "2021-05-20T17:13:00.391224Z"
  description: "Presented review"
  id: "22249e15-3161-493e-a970-c28d353837e8"
  image_id: "0a53f26bab39e6ff8c3a408f9ba681531447f16b814125f893bed954b"
  text: "Full march report"
  title: "March report"
  updated_at: "2021-05-20T17:13:00.39144Z"
}
```

Листинг 1 — Пример JSON body ответа на запрос GET /api/news/:id

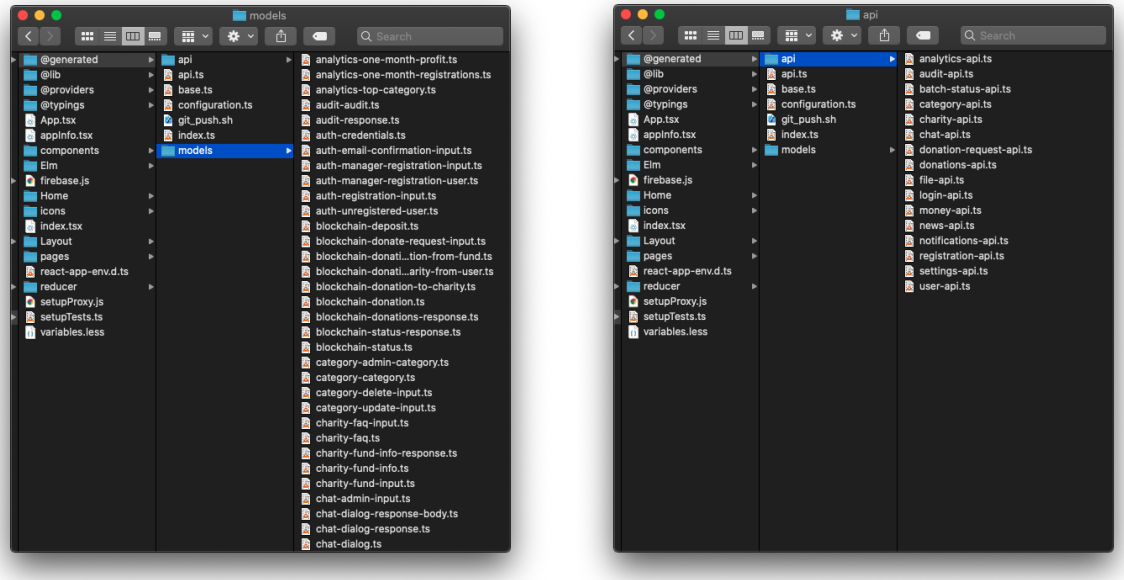

Рисунок 15 — Скриншоты папки сгенерированного кода запросов

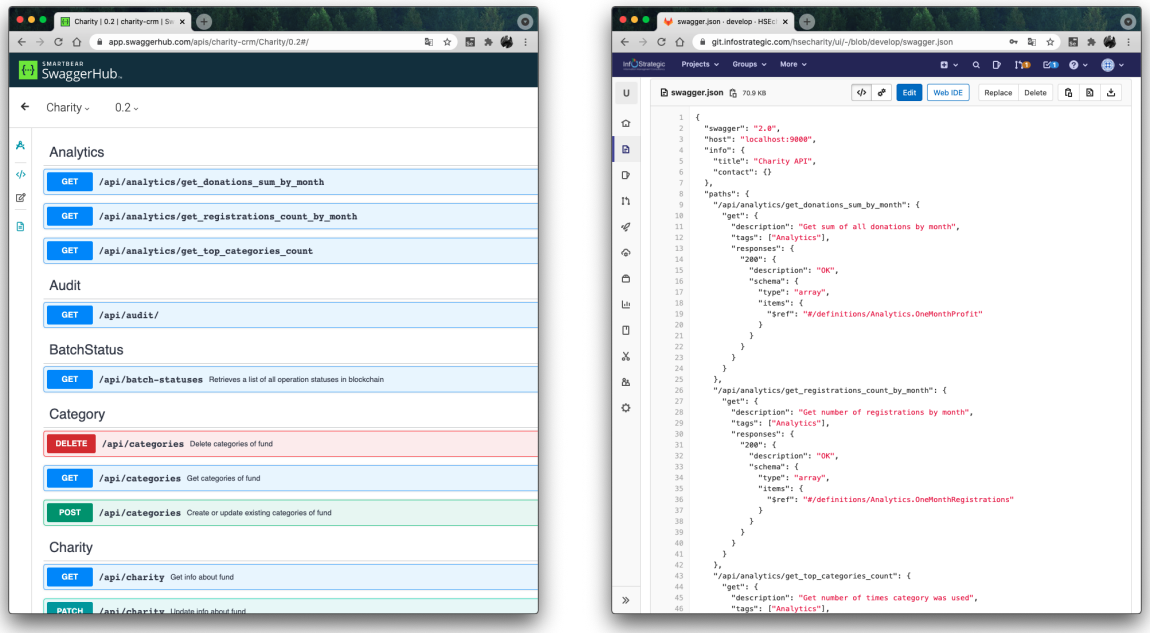

Рисунок 16 — Скриншоты документации Swagger

Одной из важных особенностей взаимодействия с API [12] является обработка статус-кодов ответа от сервера. При проектировании Web-приложении была составлена таблица со всеми возможными статус-кодами от сервера и их значениями. Данная таблица (см. таблицу 9) использовалась при обработке ответов от сервера.

Таблица 9 — Статус-коды ответов от сервера

Код	Описание	Обработка
200	Успешный ответ	В случае успешного ответа обрабатываем полученное тело ответа и демонстрируем его в UI
400	Плохие входные данные	В случае некорректного введенного и отправленного значения высвечивается уведомление на UI
401	Пользователь не авторизован	В случае получения данной ошибки пользователь не авторизован, либо закончился срок действия авторизационного токена. Либо отображается страница с авторизационной формой, либо перезапрашивается авторизационный токен через эндпойнт `refresh`
403	Недостаточно прав доступа	В случае получения данной ошибки пользователь перешел на url, к которому у него нет доступа. В данном случае высвечивается соответствующая ошибка в виде уведомления в UI

Продолжение на следующей странице

Таблица 9 – продолжение

Код	Описание	Обработка
404	Запрашиваемый контент не найден	В случае получения данной ошибки либо url запроса некорретный, либо в базе данных не найдена сущность по указанным в запросе параметрам. В данной ситуации обработка подразумевает демонстрацию ошибки на UI
500	Ошибка сервера	Данная ошибка демонстрируется в виде нотификации на UI

3.4 Дизайн модель

При проектировании дизайн модели Web-приложения были использованы стандарты UML [6].

На этапе проектирования была разработана дизайн модель, отражающая разбиение архитектуры приложения на слои. Слоистая архитектура - очень распространенный паттерн при проектировании клиент-серверных приложений, так как слой в слой бизнес-логики можно вывести клиент-серверное взаимодействие, тем самым не затрагивая слой UI компонент (см. диаграмму 17).

Рассмотрим подробней содержание и взаимодействие слоев Web-приложения.

– UI – слой с компонентами интерфейса, содержит в себе следующие подсистемы: Login (компоненты для авторизации), Transaction (компоненты, использующиеся на эранах пожертвований), Application (компоненты, использующиеся на эранах заявок), Fund (компоненты, использующиеся на эранах информации о фонде), Settings (компоненты, использующиеся на эранах личного профиля), Manager (компоненты, использующиеся на эранах инорфмации о менеджерах) и т.д; Также в подсистемы слоя интерфейса входили такие подсистемы как Layout и Elm. В Layout содержатся пакеты с макетом общих частей всех экранов приложений (заголовок, футтер, навигационная панель), в Elm подсистеме содержатся компоненты, реализованные на языке Elm с помощью одноименной архитектуры;

– BusinessLogic – слой с бизнес логикой взаимодействия с API; также этот слой содержит бизнес логику, ответственную за переходы статусов и отрисовку соответствующих компонент; в данном слое содержится также подсистема по управлению Role Based Access Controll [14], т.е ролями пользователей приложения;

– Frameworks – данный слой подробно описан в разделе 3.1.3;

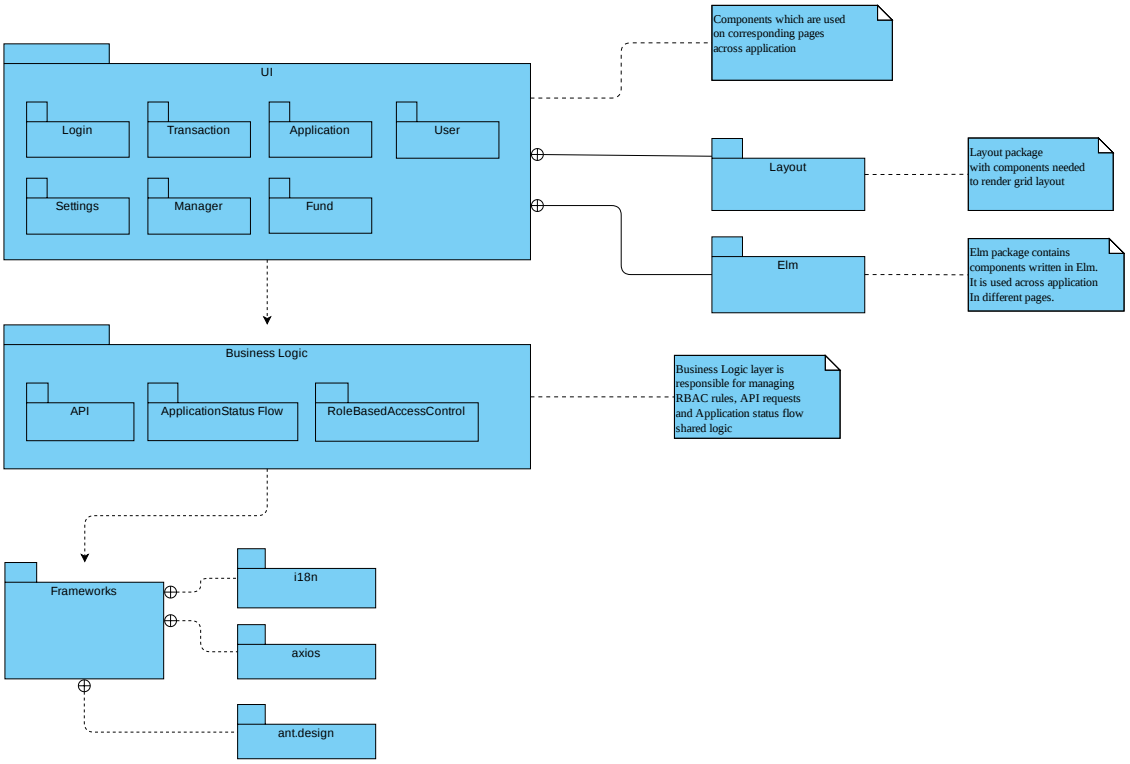

Рисунок 17 — Дизайн модель

3.5 Диаграмма компонентов

При проектировании диаграммы компонентов Web-приложения были использованы стандарты UML [6].

Так как диаграмма компонентов показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами, то в качестве физических компонентов были выбраны пакеты, содержащие файлы с расширением `tsx` и `elm`. Рассмотрим каждый слой диаграммы, изображенной на рисунке 18, подробнее.

Как уже было упомянуто в разделе 3.1.3, при разработке сервиса для отправки API запросов использовалась библиотека для кодогенерации OpenAPI Generator [24] по схеме API Swagger [12]. Таким образом все интерфейсы и модели в пакете API соответствуют спецификации API запросов. Далее на диаграмме 18 можно увидеть слой `Components`, состоящий из многочисленных подпакетов с компонентами, а также содержащий две подсистемы `Layout` и `Elm`. Первая подсистема ответственна за структурное расположения UI элементов каждой страницы приложения: в ней содержится логика по управлению хэдером, футтером, а также левым навигационным меню, присутствующим на каждой странице приложения. Вторая подсистема - Elm - содержит пакеты с компонентами, написанными на языке Elm. При рассмотрении последнего слоя диаграммы компонентов `pages` можно увидеть, что каждый подпакет слоя соответствует отдельной web странице, которая отрисовывается на UI при переходе по заданному url. Этот слой активно использует компоненты из слоя `Components`, а также API запросы из слоя `API`.

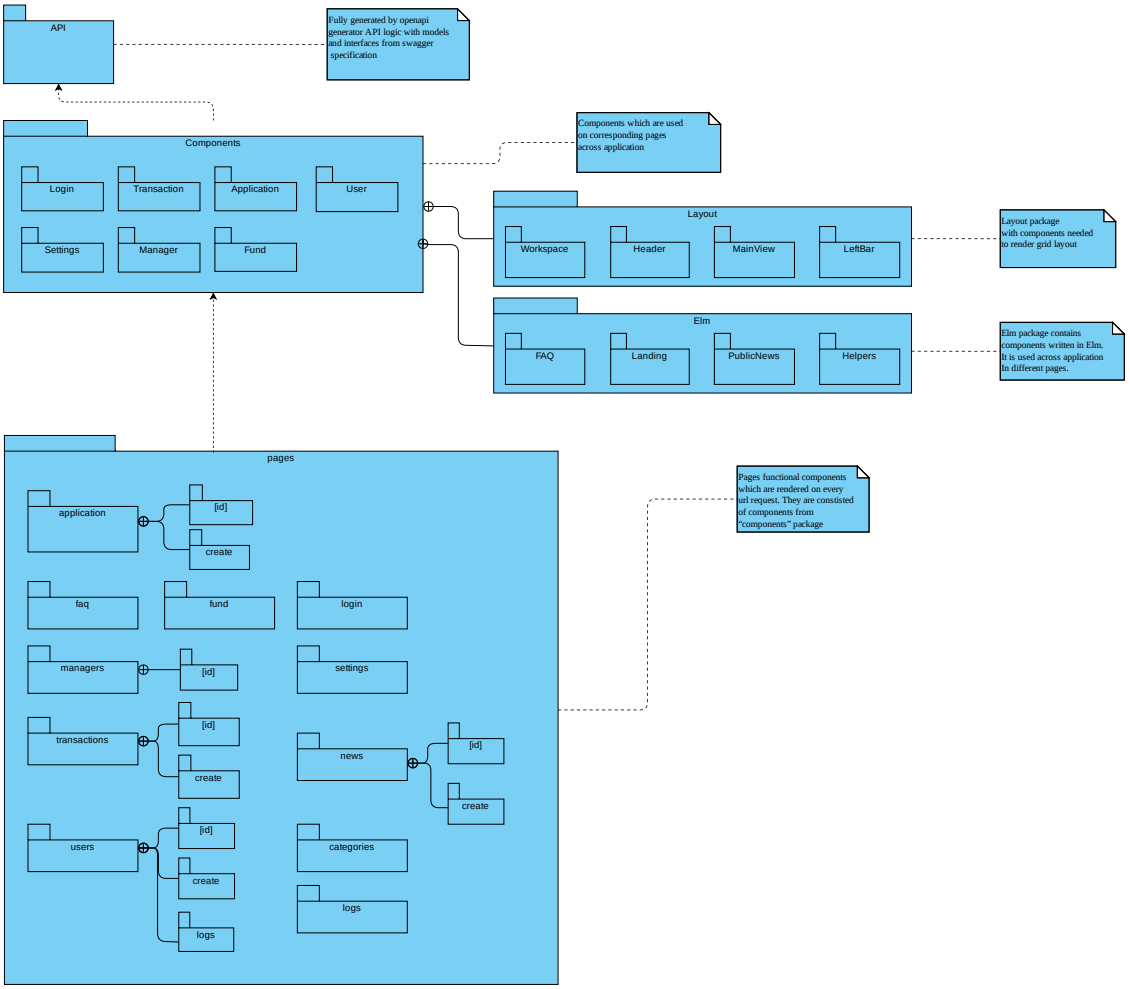

Рисунок 18 — Диаграмма компонентов

3.6 Диаграмма развертывания

При проектировании диаграммы развертывания Web-приложения были использованы стандарты UML [6].

Перед разработкой программы были составлены требования к информационной и программной совместимости, а также функциональные требования (см. Приложение А). Руководствуясь этими требованиями была составлена диаграмма развертывания, изображенная на рисунке 19. Компоненты, отмеченные белым цветом на диаграмме, `Charity CRM Backend` и `Blockchain Server` были разработаны К. И. Манежиным и И. И. Костюченко соответственно. Так как оба сервиса нужны для корректной работы клиент-серверного приложения, то они отображены на диаграмме развертывания.

Для развертывания клиент-серверного приложения использовалась Host VM - виртуальная машина, предоставленная научным руководителем. Как можно увидеть на диаграмме 19, на персональном компьютере взаимодействие с программой происходит через браузер с поддержкой HTML5 посредством канала связи HTTP непосредственно с Host VM. Как можно увидеть, на самой хостовой машине находится компонент `Web frontend Server`, который содержит внутри NGINX - компонент проксирования запросов и ReactAPP - само разработанное приложение. Через NGINX проксируются запросы на вышеупомянутый сервер `Charity CRM Backend` по API [12].

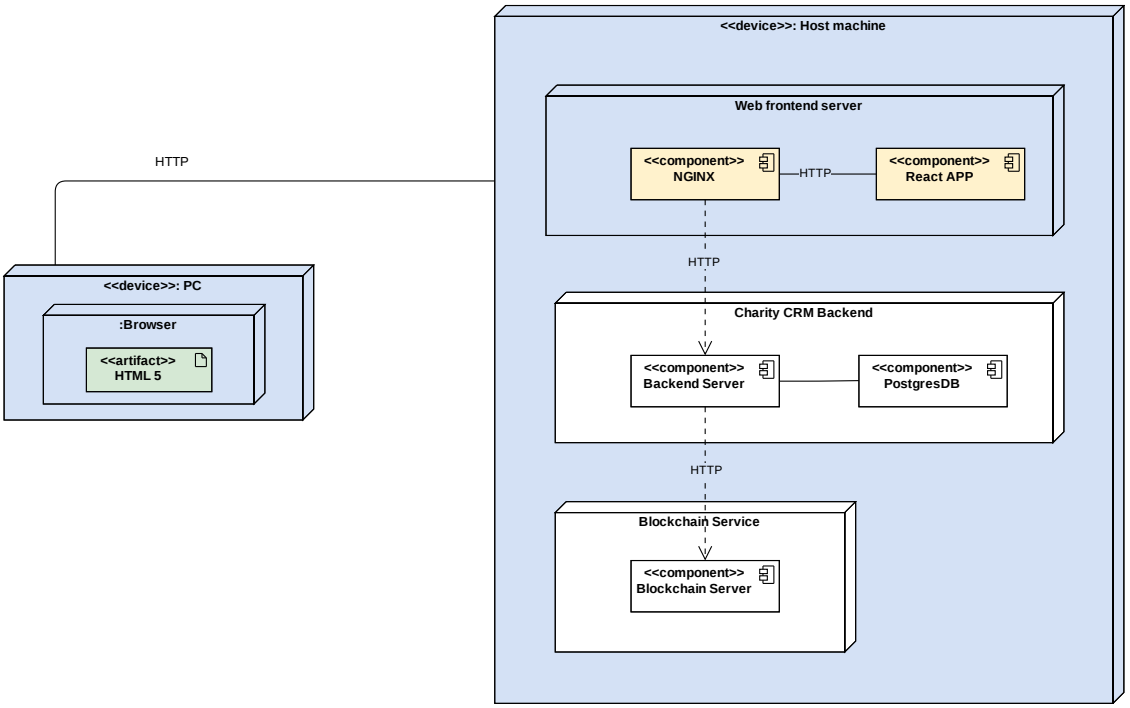

Рисунок 19 — Диаграмма развертывания

Для автоматизации развертывания стабильных версий приложения был настроен CI/CD pipeline в Gitlab репозитории, состоящий из следующих шагов: eslint, prettier (линтеры кода), test (unit-тесты), build (сборка проекта) и deploy. Первые 4 шага должны быть пройдены успешно, прежде чем начался шаг deploy: сборка и развертывание проекта с помощью Docker контейнеров. Таким образом развертывание на хост-машине происходило автоматически при мердже в ветку develop, тест-стенд доступен по адресу `https://charity.infostrategic.com`. Также для удобства тестирования стабильной версии заказчиком был развернут master стенд на по адресу `https://charity2.infostrategic.com`.

3.7 Диаграммы последовательности

3.7.1 Обработка заявки

При составлении диаграммы последовательности бизнес процесса обработки заявок (представлен в разделе 2.3 были выявлены следующие акторы: `Donee` - нуждающийся, пользователь мобильного приложения и `Comittee` - член комиссии, сотрудник фонда и пользователь Web-интерфейса. Нуждающийся через интерфейс `Android UI` взаимодействует с `ApplicationAPI`, а член комиссии через интерфейс `WebUI` также взаимодействует с `ApplicationAPI`. В свою очередь `ApplicationAPI` (API с эндпойнтами обработки заявок) взаимодействует с `ApplicationDBO` (класс, представляющий взаимодействие с базой данных) и `Blockchain module` (модуль блокчейна).

Последовательность обработки заявки инициирует нуждающийся (Donee), который через интерфейс `AndroidUI` заполняет форму новой заявки и отправляет ее на обработку сотрудникам фонда посредством отправки запроса к `ApplicationAPI`. Член комисии фонда через Web-интерфейс `WebUI` после занесения заявки в базу данных видит ее в своем интерфейсе и берет заявку в обработку (переводит с помощью UI заявку в статус «В обработке»). Это действие регистрируется в базе данных через `ApplicationAPI`. После

валидации данных заявки, если члена комисии все устраивает, то он отправляет заявку на подтверждение комисии (т.е переводит ее в новый статус «Ждет подтверждения комисии»). Изменения также регистрируются в базе данных посредством совершения запроса от `WebUI` к `ApplicationAPI`. После процесса голосования (данная диаграмма последовательности рассматривается в разделе 3.7.2) менеджер переводит заявку в следующий статус «На подтверждении пользователя». Нудающийся в свою очередь своим действием через `AndroidUI` активирует заявку, что фиксируется в системе блокчейн.

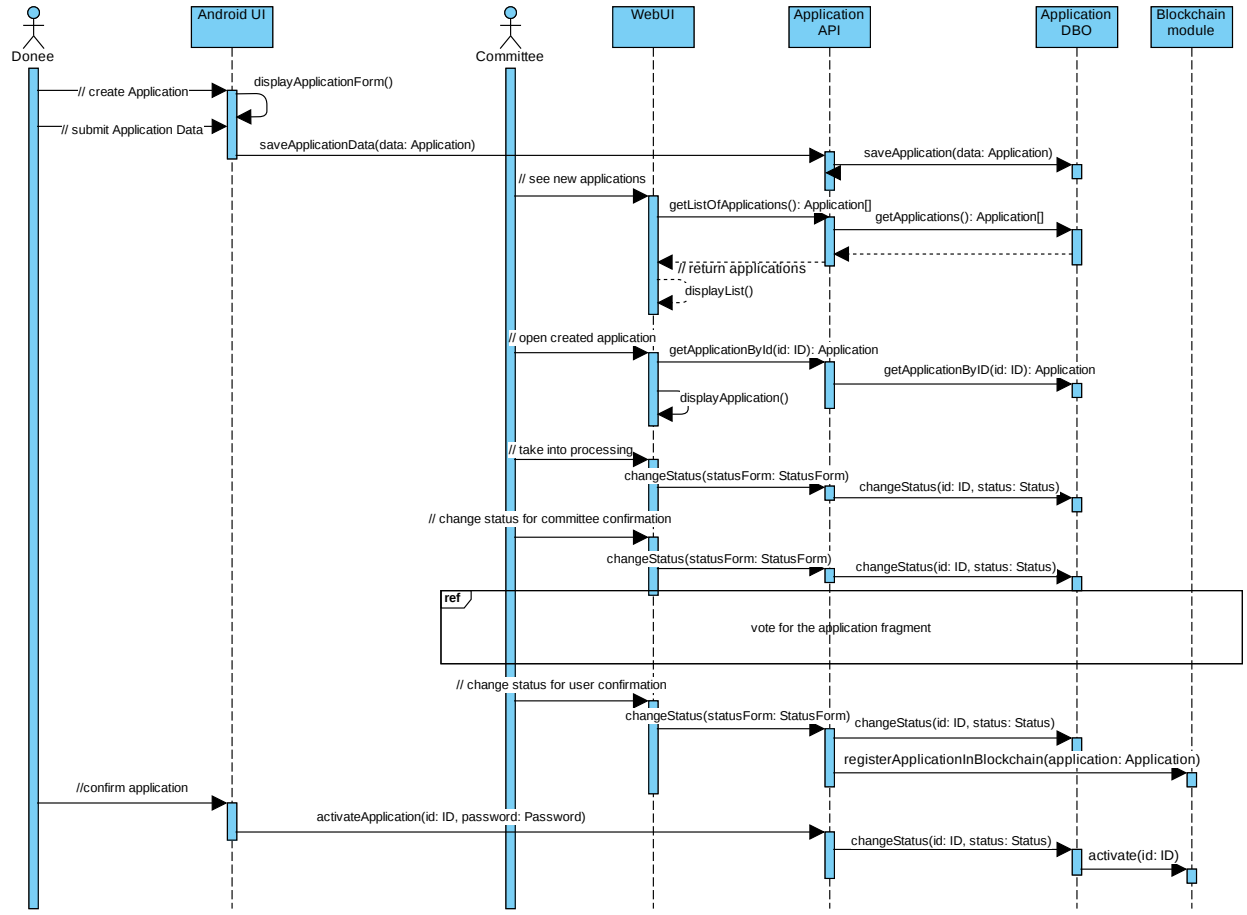

Рисунок 20 — Диаграмма последовательности обработки заявки

3.7.2 Голосование по заявке

При составлении диаграммы последовательности бизнес процесса обработки заявок (представлен в разделе 2.4 были выявлены следующие акторы: Committee - член комиссии фонда; Член комиссии через Web-интерфейс отправляет запросы в `API`, которое регистрирует изменения в базе данных (представлена паттерном DAO - Data Base Object, `VoteDAO`, `ApplicationDAO`).

Последовательность голосования по заявке инициирует член комиссии, открывая карточку заявки в статусе «Ждет подтверждения комисии». Именно по заявке в таком статусе может быть совершено голосование и принято решение по активации заявки. После открытия карточки заявки происходит проверка: может ли член комиссии голосовать по данной заявке. Если назначенные категории члена комиссии совпадают с категорией заявки, то он может принимать решение по активации данной заявки. В противном случае член комиссии может только увидеть статистику голосов других сотрудников. Если же все-таки у члена комиссии есть право отдать голос за заявку, то

он голосует с помощью UI и его голос регистрируется в системе. После того, как все ревьюеры проголосуют за или против заявки, ее уже можно будет перевести в статус «На подтверждении пользователя» или «Отказано».

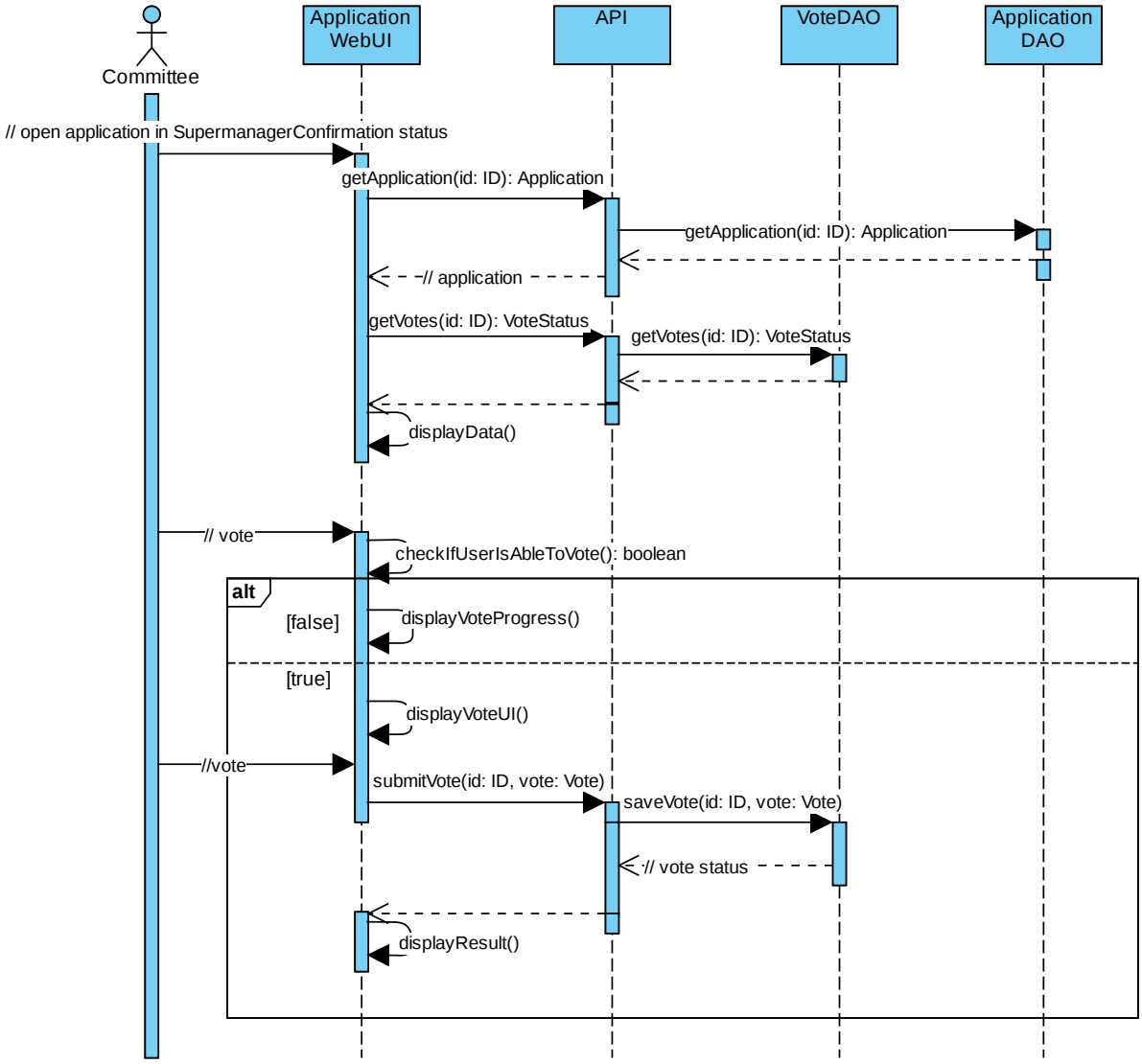

Рисунок 21 — Диаграмма последовательности голосования по заявке

3.8 Реализация ролевой модели

Как было упомянуто ранее в разделе 2.7, разарабатываемое приложение включает в себя интерфейс и логику взаимодействия для пользователей с разными уровнями доступа - сотрудниками фонда.

Пользователи клиентского Web-приложения могут иметь следующие роли в системе:

– Admin - роль для администратора фонда;

– Supermanager - роль для члена комиссии фонда;

– Manager - роль для менеджера фонда;

– Operator - роль для оператора чата поддержки фонда;

– Content-manager - роль для менеджера контента фонда;

– Visitor - неавторизованный посетитель сайта;

Более подробное описание должностей сотрудников фонда представлено в приложении B.

При проектировании реализации ролевой модели в Web-приложении использовался подход `Role-Based Access Control` [14], описанный в разделе 2.7.

Первым шагом в реализации подхода была составлена таблица (см. Приложение C) разделения возможностей, соответствующая функциональным требованиям (см. Приложение A) и прецедентам, описанным в разделе 2.1. По данной таблице был составлен следующий словарь, отображающий маппинг вышеперечисленных ролей на доступный функционал (см. Приложение D). Для того, чтобы определить, может ли пользовать с данной ролью получить доступ к странице или совершить какое-то действие, требующее особых прав, была разработан обработчик 2 для проверки уровня доступа к функциям.

```
export function check(role: Role, action: string): boolean {
  const permissions = rules[role];
  if (!permissions) {
    // role is not present in the rules
    return false;
  }

  const staticPermissions = permissions.static;

  if (staticPermissions && staticPermissions.includes(action)) {
    // static rule not provided for action
    return true;
  }

  return false;
}
```

Листинг 2 — Check функция для проверки уровня доступа

Далее был разработан React компонент (см. листинг 3), который на осбнове обработчика `check` (см. листинг 2) принимает решение какой из компонентов рендерить на UI.

```
import { FC } from "react";
import { check, Role } from "@providers/rbac-rules";

type RoleSwitchProps = {
  role: Role;
  perform: string;
  yes?: () => JSX.Element;
  no?: () => JSX.Element;
};

const RoleSwitch: FC<RoleSwitchProps> = ({
  role,
  perform,
  yes = () => null,
  no = () => null,
}) => (check(role, perform) ? yes() : no());

export default RoleSwitch;
```

Листинг 3 — RoleSwitch React компонент

3.9 Используемые паттерны

Приложение разрабатывалось с использованием технологии Single Page Application(далее - SPA)[27]. Ключевой особенностью является отсутствие перезагрузок страницы. То есть приложение напоминает нативное приложение под одну из операционных систем. Весь исполняемый код хранится у клиента, а не подгружается со стороннего сервера по необходимости (Server Side Rendering).

В процессе разработки WebUI для CRM Charity были использованы следующие паттерны проектирования:

– Singleton — шаблон проектирования создания, который использовался для инициализации инстансов Axios[22] (библиотека для выполнения запросов). На диаграмме 22 изображен пример использования данного паттерна в проекте.

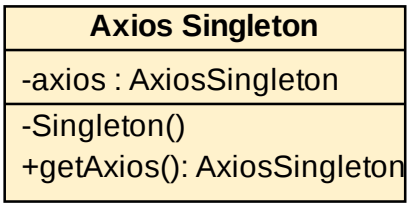

Рисунок 22 — Пример использования паттерна «Singleton»

– Factory — шаблон проектирования создания, использовался при кодогенерации запросов к API с помощью спецификации openapi[24] и документации swagger[12]. На диаграмме 23 изображен пример использования данного паттерна в проекте.

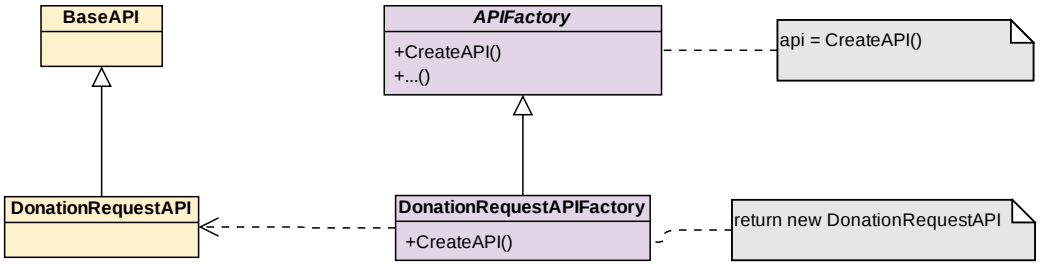

Рисунок 23 — Пример использования паттерна «Factory»

– State — поведенческий шаблон проектирования, который позволяет объекту изменять свое поведение при изменении его внутреннего состояния. Данный шаблон использовался часто в разработке компонент web-приложения, так как React framework имеет встроенный механизм работы с State в компонентах (для классовых components уже имеется встроенные `this.state`, `this.setState`, для `Factional Components` можно использовать механизм `React-hooks`[7], а именно `useState hook`[8]).

– MVU - структурный шаблон проектирования, который использовался при разработке Elm части Web-приложения. Разработка Web-страницы или компонента, написанного на Elm, делалилась на три части: model - состояние приложения, view - способ отрисовки в HTML, update - обновление состояния, основанное на полчении и отправке сообщений. На диаграмме 24 изображен базовый пример использования паттерна в приложении: Html - отрисовка нового изображения согласно полученному update (Msg), а на экране комьютера можно видеть состояние модели в данный момент.

Выводы по главе

При проектировании решения был проведен глубокий анализ технологий (представлен в разделе 3.1): были выбраны языки разработки Elm и Typescript (подробнее в разделе 3.1.1), а также фреймворк React для написания Web-приложения. Во время этапа проектирования были выбраны также библиотеки для разработки как для стека Elm, так и React+Typescript (подробнее в разделе 3.1.3). При проектировании приложения была разработана дизайн модель 17 и диаграмма компонентов 18, отображающие структуру разрабатываемого приложения.

[7] `https://reactjs.org/docs/hooks-overview.html`
[8] `https://reactjs.org/docs/hooks-state.html`

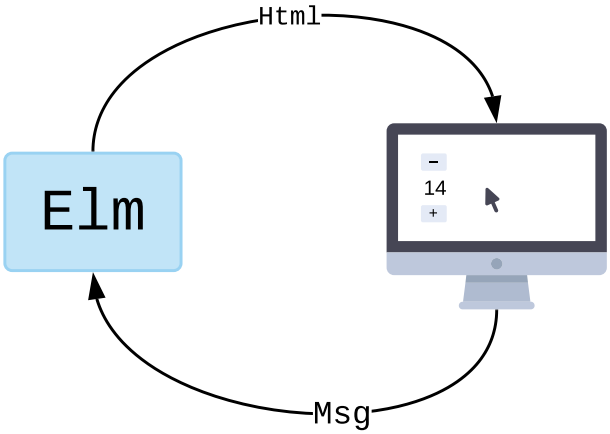

Рисунок 24 — Базовый шаблон проектирования Elm приложений

Заключение

По итогам данной работы было написано Web-приложение для сотрудников фонда «AIAIN», которое является частью CRM-системы для вышеупомянутого фонда. Весь заявленный функционал был реализован, автоматизация необходимых бизнес-процессов обеспечена.

В рамках данной работы был проведен подробный анализ существующих аналогов, выявлены сильные и слабые стороны решений на рынке. После проведенного анализа и сбора требований заказчика были составлены функциональные требования, в которых был отражен весь необходимый функционал для Web-приложения.

Перед разработкой был проведен этап анализа и проектирования, во время которых были построены диаграммы прецедентов и предметной области, а также выстроены ключевые бизнес-процессы по обработке заявок, голосованию и совершению сбора закята.

Одной из сильних сторон реализованого приложения является объединение и реализация ключевых бизнес-процессов в одном месте. Разработанный Web-интерфейс позволил объединить как администрирование пользователей, так и функциональность чатов поддержки, обработки заявок и управление контентов фонда – и все в одном месте. Каждый сотрудник фонда теперь сможет пользоваться данной системой, что позволит сэкономить время и автоматизировать работу.

В будущем развитии проекта необходимо подключить банковскую систему для совершения платежей, переводов денежных средств. Еще одним вектором развития приложения является интернализация веб-интерфейса на арабский язык. Также в будущем планируется добавить следующий функционал: создание заявок от пользователей через Web-интерфейс, расширение аналитики.

Данная работа была представлена на конференции CoCoS 2021, 16.04.2021. Работа заняла 2 место в прикладном треке [28].

Список источников

[1] Digital population worldwide URL:ttps://www.statista.com/statistics/617136/digital-population-worldwide/ [Электронный ресурс] (Дата обращения: 16.11.2020, режим доступа: свободный)

[2] Saleh H., Sergey Avdoshin, Azamat Dzhonov. Platform for Tracking Donations of Charitable Foundations based on Blockchain Technology, in: Actual Problems of Systems and Software Engineering APSSE 2019 (Invited Papers). Los Alamitos, Washington, Tokyo : IEEE Computer Society, 2019. P. 182-187 [Электронный ресурс] URL:https://ieeexplore.ieee.org/document/8943788 (Дата обращения: 16.11.2020, режим доступа: свободный)

[3] How to attract donors? URL:https://www.entrepreneur.com/article/233106 (Дата обращения: 16.11.2020, режим доступа: свободный)

[4] How to identify your competitors? - ONCE Interactive [Электронный ресурс] URL:https://onceinteractive.com/blog/how-to-identify-your-competitors/ (Дата обращения: 26.04.2021, режим доступа: свободный)

[5] Fundraising magazine crm survey, 2020 [Электронный ресурс] URL: https://www.beaconcrm.org/offer/fundraising-magazine-crm-survey-2020 (Дата обращения: 16.04.2021, режим доступа: свободный)

[6] ГОСТ Р ИСО 15745-1-2014 URL:https://docs.cntd.ru/document/1200119214 (Дата обращения: 26.04.2021, режим доступа: свободный)

[7] Единая система программной документации – М.: ИПК, Издательство стандартов, 2000, 125 стр.

[8] LMS [Электронный ресурс] URL: https://lms.hse.ru (Дата обращения: 16.11.2020, режим доступа: свободный)

[9] JSON [Электронный ресурс] URL: https://www.json.org (Дата обращения: 16.11.2020, режим доступа: свободный)

[10] Top 7 Languages for Web App Development [Электронный ресурс] URL: https://fortyseven47.com/news/top-7-languages-for-web-app-development/ (Дата обращения: 16.04.2021, режим доступа: свободный)

[11] Markdown Guide URL: https://www.markdownguide.org (Дата обращения: 16.04.2021).

[12] Swagger Charity API, v0.2 [Электронный ресурс] (Дата обращения: 31.05.2021, режим доступа: свободный) URL:https://app.swaggerhub.com/apis/charity-crm/Charity/0.2

[13] The best Web-application development languages in 2021 [Электронный ресурс] (Дата обращения: 31.05.2021, режим доступа: свободный) URL:https://medium.com/@inverita/the-best-web-application-development-languages-in-2021-6b6eb5944925

[14] Role-Based Access Control, Auth0 [Электронный ресурс] (Дата обращения: 31.05.2021, режим доступа: свободный) URL:https://auth0.com/docs/authorization/rbac/#Handling-Authorization-in-React-Apps--the-Naive-Way

[15] React-Angular-Elm-Ember performance comparison [Электронный ресурс] (Дата обращения: 31.05.2021, режим доступа: свободный) URL:https://github.com/evancz/react-angular-ember-elm-performance-comparison/blob/master/readme.md

[16] Elm lang - small assets without the headache [Электронный ресурс] (Дата обращения: 31.05.2021, режим доступа: свободный) URL:https://elm-lang.org/news/small-assets-without-the-headache

[17] Elm lang - Ports [Электронный ресурс] (Дата обращения: 31.05.2021, режим доступа: свободный) URL:https://guide.elm-lang.org/interop/ports.html

[18] RealWorldApp - Typescript [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL: https://codebase.show/projects/realworld?category=frontend&language=typescript

[19] A RealWorld Comparison 2020 [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL: https://medium.com/dailyjs/a-realworld-comparison-of-front-end-frameworks-2020-4e50655fe4c1

[20] Business Process Model and Notation - Wikipedia [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL:https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation

[21] State of JS 2020 [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL:https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/

[22] Axios - Promise based library [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL:https://github.com/axios/axios

[23] Curi Router - Documentation [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL:https://curi.js.org

[24] OpenAPI - Codegen [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL:https://github.com/OpenAPITools/openapi-generator

[25] REST - Wikipedia [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL:https://en.wikipedia.org/wiki/Representational_state_transfer

[26] SwaggerHub - Swagger API [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL:https://app.swaggerhub.com/search

[27] SPA (Single-page application), MDN Web Docs [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL:https://developer.mozilla.org/en-US/docs/Glossary/SPA

[28] CoCoS 2021 - Дипломанты конференции [Электронный ресурс] (Дата обращения 31.05.2021, режим доступа: свободный) URL:https://cs.hse.ru/studconf/2021/winners

ПРИЛОЖЕНИЕ А

Техническое задание

Представлено отдельным документом «Техническое задание. CRM-система для благотворительного фонда «AIAIN». Web-приложение для сотрудников фонда ».

ПРИЛОЖЕНИЕ В

Роли сотрудников фонда

Администратор – это сотрудник фонда, который имеет доступ к функционалу по управлению пользователями системы.

Член комиссии – у каждого фонда есть комиссия, которая принимает итоговое решение об активации заявок от нуждающихся. В обязанности членов комиссии также входит обработка заявок, просмотр пожертвований поступающих от доноров, создание категорий для заявок и так далее.

Контент-менеджер – отвечает за управление контентом фонда, а именно: часто задаваемыми вопросами, основной информацией фонда и новостями фонда;

Менеджер – большая часть сотрудников фонда состоит из менеджеров фонда. В их обязанности входит только обработка заявок, коммуникация с пользователями и сбор необходимых документов если требуется;

Оператор – оператор фонда отвечает на вопросы пользователей мобильного приложения;

Role-based access control

Таблица 10 — Таблица с распределением возможностей пользователей по ролям

Действие	Описание	Admin	S.manager	Manager	Operator	C.manager	Visitor
auth:login	Авторизация	НЕТ	НЕТ	НЕТ	НЕТ	НЕТ	ДА
faq:pretty	Просмотр FAQ	НЕТ	ДА	НЕТ	НЕТ	ДА	ДА
faq:edit	Изменение FAQ	НЕТ	НЕТ	НЕТ	НЕТ	ДА	НЕТ
fund:edit	Изменение информации о фонде	НЕТ	НЕТ	НЕТ	НЕТ	ДА	НЕТ
fund:description	Просмотр информации о фонде	НЕТ	ДА	НЕТ	НЕТ	ДА	ДА
fund:index	Просмотр аналитики о фонде	НЕТ	ДА	НЕТ	НЕТ	НЕТ	НЕТ
news:index	Просмотр новостей о фонде	НЕТ	НЕТ	НЕТ	НЕТ	ДА	НЕТ
news:edit	Изменение новостей о фонде	НЕТ	НЕТ	НЕТ	НЕТ	ДА	НЕТ
news:create	Создание новостей о фонде	НЕТ	НЕТ	НЕТ	НЕТ	ДА	НЕТ
applications: show	Просмотр заявки	НЕТ	ДА	ДА	НЕТ	ДА	НЕТ
applications: create	Создание заявки	НЕТ	ДА	ДА	НЕТ	НЕТ	НЕТ
applications: index	Просмотр списка заявок	НЕТ	ДА	ДА	НЕТ	НЕТ	НЕТ
applications: edit	Редактирование заявки	НЕТ	ДА	ДА	НЕТ	НЕТ	НЕТ
applications: can-vote	Голосование за активацию заявки	НЕТ	ДА	НЕТ	НЕТ	НЕТ	НЕТ
categories: index	Проосмотр, редактирование категорий	НЕТ	ДА	НЕТ	НЕТ	НЕТ	НЕТ
transactions: index	Список транзакций	НЕТ	ДА	НЕТ	НЕТ	НЕТ	НЕТ
transactions: show	Просмотр информации о транзакции	НЕТ	ДА	НЕТ	НЕТ	НЕТ	НЕТ
transactions: create	Создание ручной транзакции	НЕТ	ДА	НЕТ	НЕТ	НЕТ	НЕТ
settings:index	Изменение личного профиля	ДА	ДА	ДА	ДА	ДА	НЕТ

Продолжение на следующей странице

Таблица 10 – продолжение

Действие	Описание	Admin	S.manager	Manager	Operator	C.manager	Visitor
settings:index	Изменение личного профиля	ДА	ДА	ДА	ДА	ДА	НЕТ
notifications:index	Просмотр уведомлений	ДА	ДА	ДА	ДА	ДА	НЕТ
users:show	Просмотр профиля пользователя	ДА	ДА	ДА	ДА	НЕТ	НЕТ
users:index	Просмотр списка пользователей системы	ДА	НЕТ	НЕТ	НЕТ	НЕТ	НЕТ
users:create	Регистрация пользователей в системе	ДА	НЕТ	НЕТ	НЕТ	НЕТ	НЕТ
users:edit	Изменение информации о пользователе	ДА	НЕТ	НЕТ	НЕТ	НЕТ	НЕТ
logs:index	Просмотр логов в системе	ДА	НЕТ	НЕТ	НЕТ	НЕТ	НЕТ
managers:index	Просмотр списка менеджеров системы	НЕТ	ДА	НЕТ	НЕТ	НЕТ	НЕТ
managers:show	Просмотр профиля менеджера	НЕТ	ДА	НЕТ	НЕТ	НЕТ	НЕТ
managers:show	Просмотр профиля менеджера	НЕТ	ДА	НЕТ	НЕТ	НЕТ	НЕТ
chat:show	Просмотр диалога с пользователем	НЕТ	НЕТ	НЕТ	НЕТ	ДА	НЕТ
chat:index	Просмотр списка диалогов с пользователем	НЕТ	НЕТ	НЕТ	НЕТ	ДА	НЕТ

Реализация RBAC, rbac-rules.ts

```
export enum Role {
  visitor = "visitor",
  manager = "manager",
  supermanager = "supermanager",
  contentManager = "contentManager",
  operator = "operator",
  admin = "admin",
}

const rules = {
  visitor: {
    static: [
      "auth:login",
      "faq:pretty",
      "fund:description-pretty",
      "news:public",
    ],
  },
  contentManager: {
    static: [
      "applications:show",
      "settings:index",
      "users:show",
      "fund:index",
      "fund:faq-index",
      "fund:description",
      "fund:description-edit",
      "faq:edit",
      "news:edit",
      "news:create",
      "news:index",
      "notifications:index",
    ],
  },
  operator: {
    static: ["chats:show", "chats:index", "settings:index"],
  },
  manager: {
    static: [
      "applications:index",
      "settings:index",
      "application:edit",
      "applications:show",
      "applications:create",
      "users:show",
      "user:view-applications",
      "notifications:index",
    ],
  },
  supermanager: {
    static: [
      "applications:index",
      "applications:show",
```

```
      "application:edit",
      "application:can-vote",
      "applications:create",
      "settings:index",
      "categories:index",
      "users:show",
      "user:view-applications",
      "fund:index",
      "fund:description",
      "transactions:show",
      "transactions:index",
      "transactions:create",
      "managers:index",
      "managers:show",
      "fund:faq-index",
      "notifications:index",
      "transactions:distribute",
    ],
  },
  admin: {
    static: [
      "users:index",
      "users:show",
      "user:edit",
      "users:create",
      "user:view-sessions",
      "user:show-admin",
      "settings:index",
      "logs:index",
      "notifications:index",
    ],
  },
};
```

Листинг 4 — Словарь маппинга ролей на функциональность