МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики и кибернетики

Кафедра программных систем

**ОТЧЕТ**  
 по лабораторному практикуму по дисциплине «Современные

методы разработки и проектирования программных комплексов»

по теме «Мобильное приложение для поиска лекарств из

открытых источников и напоминания об их приеме»

Обучающийся группы 6132-020402D К.А. Портнов

Обучающийся группы 6132-020402D В.И. Хорина

Руководитель Л.С. Зеленко

Самара 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики и кибернетики

Кафедра программных систем

ЗАДАНИЕ

на лабораторный практикум по дисциплине

«Современные методы разработки и проектирования   
программных комплексов»

обучающимся в группе № 6132-020402D

К.А. Портнову

В.И. Хориной

1. Тема проекта:«Мобильное приложение для поиска лекарств из открытых источников и напоминания об их приеме»
2. Исходные данные к проекту**:** см. приложение к заданию
3. Перечень вопросов, подлежащих разработке:
   1. Произвести анализ предметной области: изучить основные принципы реализации лекарственных препаратов, изучить особенности их приема
   2. Выполнить обзор существующих систем-аналогов
   3. Разработать информационно-логический проект системы
   4. Разработать и реализовать программное и информационное обеспечение, провести его тестирование и отладку
   5. Оформить документацию
   6. Подготовить презентацию по разработанной системе
4. Перечень графических разработок:
   1. Структурная схема системы
   2. Диаграмма вариантов использования и диаграмма классов
   3. Схемы основных алгоритмов
5. Календарный план выполнения работ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Содержание работы по этапам | Объем этапа в % к общему объему проекта | Срок  окончания | Фактическое выполнение |
| 1 | Оформление технического задания и его утверждение | 5 | 04.10.2022 |  |
| 2 | Описание и анализ предметной области | 10 | 04.10.2022 |  |
| 3 | Проектирование системы | 30 | 15.11.2022 |  |
| 3.1 | Разработка структурной схемы системы | 5 | 18.10.2022 |  |
| 3.2 | Разработка прототипа интерфейса пользователя | 10 | 25.10.2022 |  |
| 3.3 | Разработка информационно-логического проекта системы и его предъявление руководителю | 15 | 15.11.2022 |  |
| 4 | Реализация проекта, разработка контрольных примеров. Предъявление реализации руководителю | 45 | 29.11.2022 |  |
| 5 | Корректировка проекта и оформление документации проекта. Защита проекта с представлением презентации. | 10 | 27.12.2022 |  |

Задание принял  
 к исполнению 14.09.2022 К.А. Портнов

14.09.2022 В.И. Хорина

ПРИЛОЖЕНИЕ  
к заданию на лабораторный практикум  
обучающимся в группе № 6132-020402D  
К.А. Портнову

В.И. Хориной

Тема проекта: «Мобильное приложение для поиска лекарств из открытых источников и напоминания об их приеме»

Исходные данные к проекту:

1. Характеристика объекта автоматизации:

## объект автоматизации: аптечная сеть;

## виды автоматизируемой деятельности:

* + процесс авторизации/регистрации пользователя;
  + процесс поиска лекарственных препаратов по заданному критерию;
  + процесс составление расписания приема лекарств;
  + процесс просмотра запланированных приемов лекарств;
  + процесс получения оповещений;

## длина номера мобильного телефона – 11 символов;

## длина кода подтверждения по СМС– 4 символа;

## количество критериев для поиска – 4;

## количество элементов для поиска по составу – 1;

1. количество элементов для поиска по стране производителю – 1;
2. количество элементов для поиска по названию – 1;
3. количество ссылок для поиска препарата в аптечных сетях – 2;
4. количество типов подписок на оповещения – 2;
5. количество типов периодичности приема лекарств – 2;
6. количество напоминаний – 1;
7. начало доступного периода просмотра напоминаний в «календаре» – начало месяца;
8. максимальное количество напоминаний одного препарата в день – 5.
9. Требования к информационному обеспечению:
10. информационное обеспечение разрабатывается на основе следующего источника:
    * Лекарственное средство [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Лекарственное\_средство (дата обращения: 08.10.2022);
    * Аптека [Электронный ресурс]. URL: https://ru.wikipedia.org/  
      wiki/Аптека (дата обращения: 08.10.2022);
11. структура базы данных разрабатывается на основании следующих сведений:
    * о пользователе (телефон, код, e-mail, имя, список лекарств, список напоминаний);
    * о лекарстве (название, сертификат, дозировка, форма выпуска, состав, страна производитель, изображение);
    * напоминания (лекарство, дата, время, дозировка, время до приема);
    * сертификат (номер, дата регистрации, статус);
    * страна производитель (название, изображение флага);
12. в системе имеются справочники:

* лекарственные препараты;
* страны производители;

1. изображения лекарств хранятся в файлах формата .png;
2. должна быть обеспечена целостность базы данных и защита от несанкционированного доступа.
3. Требования к техническому обеспечению:
   1. Требования к техническому обеспечению серверной части системы:
4. тип ЭВМ – IBM PC совместимый;
5. объем ОЗУ – не менее 2 Гб;
6. объем свободного пространства на внешнем диске – не менее 50 Гб;
7. наличие подключения к сети Интернет;
8. технические характеристики определяются в процессе выполнения проекта.
   1. Требования к техническому обеспечению клиентской части системы:
9. тип ЭВМ – мобильное устройство Android/IOS;
10. экран с разрешающей способностью не ниже 1080 х 2340;
11. манипулятор – сенсорный экран;
12. наличие подключения к сети Интернет;
13. технические характеристики определяются в процессе выполнения проекта.
14. Требования к программному обеспечению:
    1. Требования к программному обеспечению серверной части системы:
15. тип операционной системы – Windows 7 и выше/Linux;
16. .NET Runtime 7.0
17. СУБД – SQL Server 15.0.
    1. Требования к программному обеспечению клиентской части системы:
18. тип операционной системы – Android 12 и выше – API 21 и выше / IOS 14.2 и выше;
    1. Требования к программному обеспечению рабочего места разработчика:
19. тип операционной системы – Windows 7 и выше, MacOS 12;
20. язык программирования – С#;
21. среда программирования – Visual Studio 2022;
22. .NET SDK 7.0;
23. .NET MAUI SDK;
24. СУБД – SQL Server 15.0;
25. среда проектирования – Visio 2016.
26. Общие требования к проектируемой системе:

5.1 Функции, реализуемые системой:

1. функции системы:
   * аутентификация пользователя в системе;
   * регистрация пользователя в системе;
   * рассылка напоминаний о приеме лекарств;
   * ускоренная авторизация;
   * отслеживание ранее отправленных уведомлений;
   * фильтрация лекарств по параметрам (название, состав, страна производитель, наличие сертификата);
   * автоматическое составление уведомлений для последующей отправки;
   * визуализация процессов работы с приложением;
   * проверка дублирования лекарств;
   * выдача справочной информации о системе;
   * выдача справочной информации о лекарственном препарате;
   * генерирование ссылок для поиска препарата в аптечных сетях;
2. функции пользователя:
   * регистрация в приложении (ввод телефона);
   * авторизация в приложении (ввод кода, отправленного по SMS);
   * возможность при повторной авторизации использовать touchId/faceId;
   * возможность по ссылке перейти в интернет магазин аптечных сетей;
   * возможность изменения своих персональных данных в личном кабинете (имя, e-mail);
   * возможность удалить свой аккаунт;
   * составление фильтров для поиска лекарств;
   * просмотр запланированных приемов;
   * отслеживание плана приема лекарства;
   * подтверждение приема лекарства;
   * добавление лекарств в свою библиотеку;
   * составление графика приема лекарств;
   * включение/отключение оповещений системы;
   * просмотр справочной информации.

5.2 Технические требования к системе:

1. режим работы ‑ диалоговый;
2. максимальная задержка получения уведомления – 5 мин;
3. максимальная задержка получения SMS – 5мин;
4. максимальное время загрузки данных на странице – 1с;
5. система должна удовлетворять санитарным правилам и нормам  
    СанПин 2.2.2./2.4.2198-07;
6. условия работы средств вычислительной техники (содержание вредных веществ, пыли и подвижность воздуха) должны соответствовать ГОСТ 12.1.005, 12.01.007;
7. температура окружающего воздуха – 15-35°С;
8. влажность воздуха – 45-75%.

Руководитель   
проекта Л.С. Зеленко

Задание принял  
к исполнению 14.09.2022 К.А. Портнов

14.09.2022 В.И. Хорина

РЕФЕРАТ

Пояснительная записка 105 с, 19 рисунков, 23 таблиц, 23 источников,  
2 приложения.

Графическая часть: ??? слайдов презентации PowerPoint.

ЛЕКАРСТВЕННЫЕ ПРЕПАРАТЫ, НАПОМИНАНИЯ О ПРИЕМЕ, PUSH-УВЕДОМЛЕНИЯ, СЛОВАРЬ ПРЕПАРАТОВ, СОСТАВЛЕНИЕ ГРАФИКА, ТАБЛЕТНИЦА, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ.

Объектом автоматизации является аптечная сеть.

Во время лабораторного практикума разработаны алгоритмы и соответствующее им мобильное приложение, позволяющее составлять графика приема лекарственных препаратов. Лекарственные препараты, как и созданный график приема хранятся в реляционной базе данных и доступ к ним осуществляется через серверную часть приложения. Пользователь через интерфейс мобильного приложения добавляет лекарственные препараты в свою библиотеку и задает периодичность, дозировку и время приема на основании чего создается график приема лекарств. Специальный алгоритм на серверной части приложения периодически проверяет срабатывание таймеров на прием лекарств и отправляет соответствующее сообщение пользователю через сервисы уведомлений Google/Apple. Пользователь имеет возможность посмотреть детальное описание лекарственного препарата и перейти по ссылке на сайты аптечных сетей.

Клиентская часть приложения написана на языке С# на платформе .NET MAUI в среде Visual Studio 2022 и функционирует под управлением операционных систем Android 12 и выше – API 21 и выше / IOS 14.2 и выше.

Серверная часть приложения написана на языке C# на платформе ASP .NET в среде Visual Studio 2022 и функционирует под управлением операционных систем Windows 7 и выше/Linux.

Доступ к данным осуществляется под управлением СУБД Microsoft SQL Server 15.0.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 13](#_Toc119614934)

[1 Описание и анализ предметной области 15](#_Toc119614935)

[1.1 Описание предметной области 15](#_Toc119614936)

[1.1.1 Основные понятия и определения 15](#_Toc119614937)

[1.1.2 Примеры лекарственных препаратов 17](#_Toc119614938)

[1.2 Описание систем-аналогов 19](#_Toc119614939)

[1.2.1 «CareClinic» 20](#_Toc119614940)

[1.2.2 «Pills Time» 21](#_Toc119614941)

[1.3 Постановка задачи 22](#_Toc119614942)

[2 Проектирование системы 27](#_Toc119614943)

[2.1 Выбор и обоснование архитектуры системы 27](#_Toc119614944)

[2.2 Структурная схема системы 29](#_Toc119614945)

[2.3 Разработка прототипа интерфейса пользователя системы 32](#_Toc119614946)

[2.4 Разработка информационно-логического проекта системы 39](#_Toc119614947)

[2.4.1 Язык UML 39](#_Toc119614948)

[2.4.2 Диаграмма вариантов использования 41](#_Toc119614949)

[2.4.3 Диаграмма классов 44](#_Toc119614950)

[2.5 Логическая модель данных 49](#_Toc119614951)

[2.6 Выбор и обоснование алгоритмов обработки данных 53](#_Toc119614952)

[2.7 Выбор и обоснование комплекса программных средств 55](#_Toc119614953)

[2.7.1 Выбор языка программирования 55](#_Toc119614954)

[2.7.2 Выбор среды разработки 55](#_Toc119614955)

[2.7.3 Выбор системы управления базами данных 56](#_Toc119614956)

[2.7.4 Выбор платформы для создания мобильных приложений 57](#_Toc119614957)

[2.7.5 Выбор поддерживаемых платформ 57](#_Toc119614958)

[Клиентская часть 57](#_Toc119614959)

[Серверная часть 58](#_Toc119614960)

[2.7.6 Выбор поставщика СМС-сообщений 58](#_Toc119614961)

[2.7.7 Выбор поставщика PUSH-уведомлений 58](#_Toc119614962)

[3 Реализация системы 60](#_Toc119614963)

[3.1 Разработка и описание интерфейса пользователя 60](#_Toc119614964)

[3.2 Диаграммы реализации 68](#_Toc119614965)

[3.2.1 Диаграмма развертывания 69](#_Toc119614966)

[3.2.2 Диаграмма классов 69](#_Toc119614967)

[3.3 Физическая модель данных 70](#_Toc119614968)

[3.4 Выбор и обоснование комплекса технических средств 77](#_Toc119614969)

[3.4.1 Расчет объема занимаемой памяти 77](#_Toc119614970)

[3.4.2 Минимальные требования, предъявляемые к системе 78](#_Toc119614971)

[ЗАКЛЮЧЕНИЕ 80](#_Toc119614972)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 81](#_Toc119614973)

[ПРИЛОЖЕНИЕ А Руководство пользователя 84](#_Toc119614974)

[А.1 Назначение системы 84](#_Toc119614975)

[А.2 Условия работы системы 84](#_Toc119614976)

[А.3 Установка системы 84](#_Toc119614977)

[А.4 Работа с системой 84](#_Toc119614978)

[А.4.1 Работа с системой в режиме администратора (если необходимо) 85](#_Toc119614979)

[А.4.2 Работа с системой в режиме пользователя 85](#_Toc119614980)

[ПРИЛОЖЕНИЕ Б Листинг модулей программы 86](#_Toc119614981)

ВВЕДЕНИЕ

В последние годы стала прослеживаться тенденция увеличения средней продолжительности жизни человека. Это связано с множеством факторов таких как: люди стали лучше следить за своим здоровьем, лечить различные болезни на ранних стадиях, когда лечение самое эффективное, улучшилось качество питания, качество жилищных условий, люди стали внимательнее относиться к уровню личной гигиены и безусловно один из главных факторов – стремительное развитие современной медицины – с каждым годом на прилавках магазинов и в медицинских учреждениях появляется все больше и больше различных лекарственных препаратов способных противостоять почти любой известной на данный момент науке болезни или инфекции.

К примеру, группа препаратов антибиотиков, вошедших в жизнь человечества более полувека назад, спасли миллионы жизней, поборов такие болезни, как пневмония, туберкулез, гангрена и прочие инфекционные заболевание, без лечения приводящие к летальному исходу.

Но в нашем мире всегда должно быть равновесие и, если мы учимся бороться с каким-либо болезнями, то они, в свою очередь, начинают эволюционировать и приобретать иммунитет к разработанному лечению, из-за чего необходимо постоянно совершенствовать лекарственные препараты. Что в свою очередь является одной из причин почти бесчисленного множества все возможных лекарственных препаратов.

Все чаще по назначению врача приходится принимать сразу одновременно несколько различных лекарств с разной периодичностью, дозировкой и продолжительностью приема. Чтобы ничего не перепутать, приходиться вести целое расписание для приема лекарств и постоянно сверяться с этим списком, чтобы ничего не забыть и не пропустить прием, что по своей сути является очень рутинной и монотонной задачей, и далеко не все могут успешно с ней справиться, но на помощь человечеству в решении этой проблемы приходят современные технологий с мобильными приложениями для телефонов, которые есть у каждого человека всегда под рукой и с системой уведомлений (push notification), позволяющей оперативно отправить сообщение на мобильное устройство.

Во время выполнения лабораторного практикума необходимо разработать мобильное приложение с функциями напоминания о необходимости приема лекарств, с помощью которой можно создавать свой список лекарств и настраивать время, дозировку и периодичность приема лекарств.

При проектировании системы будут использоваться методологии структурного проектирования систем и ООАП (Object-Oriented Analysis/Design), в основу которой положена объектно-ориентированная методология представления предметной области в виде объектов, а также язык моделирования UML (Unified Modeling Language), который является стандартным инструментом для разработки «чертежей» программного обеспечения [1].

1. Описание и анализ предметной области

Предметная область – часть реального мира, которая имеет существенное значение или непосредственное отношение к процессу функционирования программы, она включает в себя только те объекты и взаимосвязи между ними, которые необходимы для описания требований и условий решения конкретной задачи [2].

* 1. Описание предметной области
     1. Основные понятия и определения

Лекарственные препарат – это лекарственное средство (действующее вещество или смесь) в определенной дозе в виде определенной лекарственной формы, готовое к применению, наиболее часто применяемое для профилактики, диагностики и лечения заболеваний. Перед употреблением в медицинской практике лекарственные средства должны проходить клинические исследования и получать разрешение к применению [3].

Наиболее распространенными формами лекарственных средств являются:

* таблетки;
* капсулы;
* капли;
* растворы;
* мази.

На рисунке 1 приведены различные формы лекарственных средств.

Сырьем для получения лекарственных средств служат:

* растения (листья, трава, цветки) и продукты их обработки (жирные и эфирные масла, соки, камеди, смолы);
* животные (железы и органы животных);
* ископаемое органическое сырье (нефть и нефти-продукты);
* неорганические ископаемые (минеральные породы);
* всевозможные органические соединения (продукты крупной химической промышленности).

Рисунок 1 – Примеры формы лекарственных средств



Государство достаточно жестко регулирует обращение лекарственных средств. Производство разрешенных лекарственных средств должно соответствовать требованиям ГОСТ Р 52249-2009. Каждое лекарственное средства для реализации на территории России должно получить регистрационное удостоверение [3].

Доза – это количество лекарственного средства, которое принимается за один раз. Оно может быть выражено как масса таблетированного препарата (например, 250 мг), объем раствора (например, 10 мл), количество капель или впрыскиваний (например, 2 капли, 1 впрыскивание) [3].

Основные разновидности доз:

* минимальная (эффект небольшой степени);
* средняя (наиболее часты в клинической практике);
* максимальная (самое эффективное терапевтическое действие);
* предельно допустимая (токсичная для организма);

Аптечная сеть – совокупность аптечных организаций (аптек), занимающихся розничной торговлей лекарственными средствами, изделиями медицинского назначения и товарами сопутствующего ассортимента с единым товарно-финансовым потоком, с централизованной системой управления, единой маркетинговой стратегией, единым имиджем [4].

Классификация по характеру деятельности:

* производственные (в производственных аптеках находятся производственные помещения, где изготавливаются лекарства, и зал для посетителей);
* аптеки готовых лекарственных форм (аптеки данного типа занимаются только реализацией лекарств, изготовленных на заводах фирм-производителей).

Основные функции аптечных организаций:

* реализация готовых лекарственных препаратов;
* изготовление лекарственных препаратов;
* реализация лекарственного растительного сырья;
* оказание первой медицинской помощи;
* оказание консультативной помощи.
  + 1. Примеры лекарственных препаратов

Ниже приведено описание лекарственного препарата “АнвиМакс” (рисунок 2). В его состав входят:

* парацетамол 360 мг;
* фскорбиновая кислота 300 мг;
* кальция глюконата моногидрат 100 мг;
* римантадина гидрохлорид 50 мг;
* рутозид 20 мг;
* лоратадин 3 мг;

  
Рисунок 2 – Лекарственный препарат «АнвиМакс»

Формы выпуска – таблетки шипучие с ароматизаторами. Производитель – общества с ограниченной ответственностью ФармВИЛАР (Россия). Регистрационное удостоверение – ЛП-№ (000151)-(РГ-RU) от 03.03.2021. Прочие характеристики:

* вид: гибридный;
* доп. признаки: многокомпонентный.

Ниже приведено описание лекарственного препарата «АнвиМакс Кидс» (рисунок 3).

  
Рисунок 3 – Лекарственный препарат «АнвиМакс Кидс»

В его состав входят:

* парацетамол;
* хлорфенамин;
* аскорбиновая кислота.

Формы выпуска – порошок для приготовления раствора для приема внутрь. Производитель – Corporations Контракт Фармакал Корпорейшн, США (Соединенные Штаты). Регистрационное удостоверение – ЛП-№ (000373)-(РГ-RU) от 28.09.2021.

Прочие характеристики:

* вид: хорошо изученный.

Ниже приведено описание описана лекарственного препарата «Апрепитант» (рисунок 4).



Рисунок 4 – Лекарственный препарат «Апрепитант»

В его состав входят:

* апрепитант.

Формы выпуска – набор капсул, без сахара. Производитель – акционерное общество АО "Биоком" (Россия). Регистрационное удостоверение – ЛП-№ (001092)-(РГ-RU) от 02.08.2022.

Прочие характеристики:

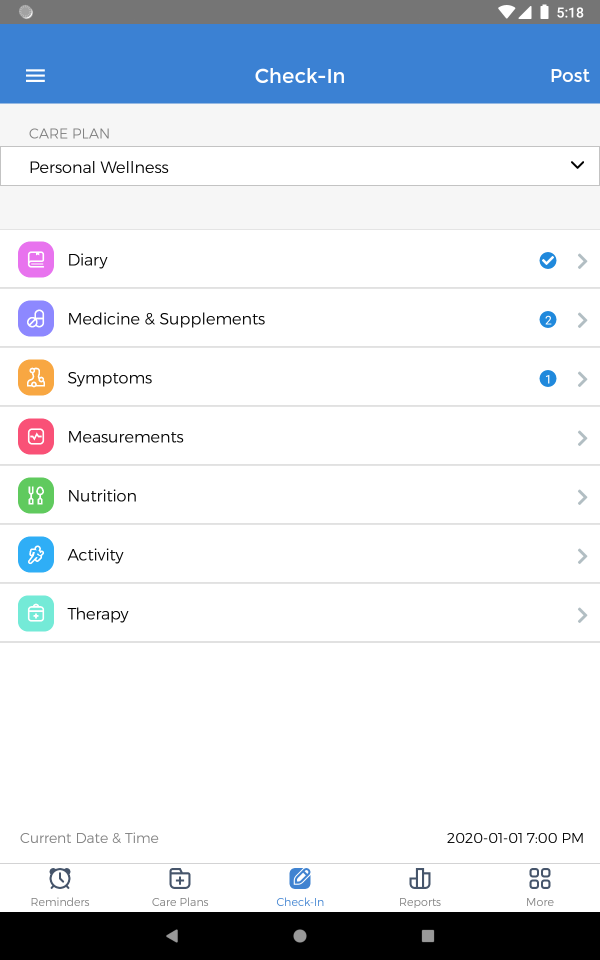
* вид: воспроизведенный.
  1. Описание систем-аналогов

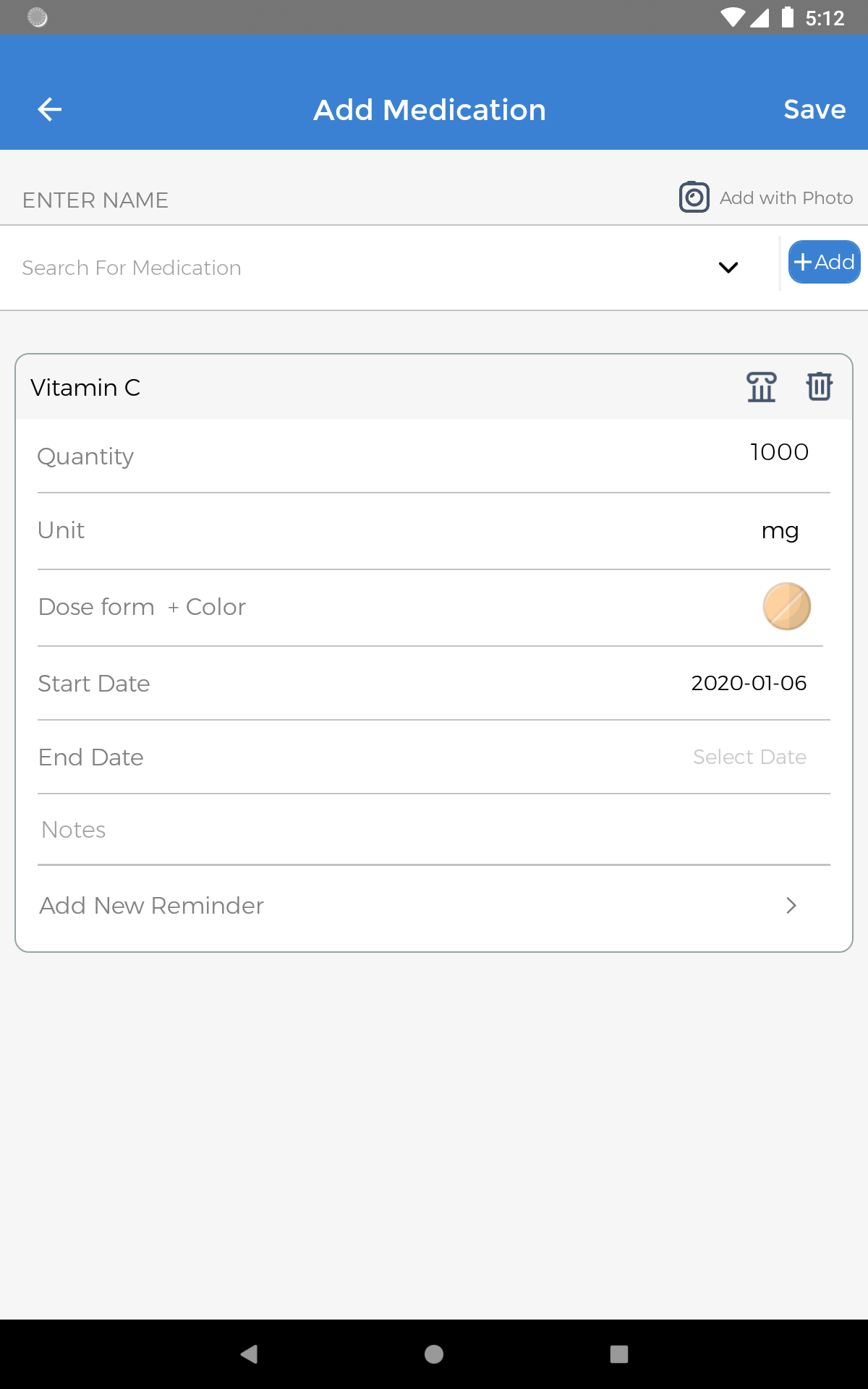
Рассмотрим возможности систем-аналогов разрабатываемого приложения.

* + 1. «CareClinic»

«CareClinic» – это приложение для отслеживания симптомов или дневника симптомов, это ежедневное универсальное приложение для здоровья и хорошего самочувствия, которое помогает измерять, изучать и улучшать свое здоровье. Данное приложение поможет никогда больше не забыть свои лекарства или таблетки. Это незаменимое приложение для отслеживания приема лекарств и напоминаний для здоровья и благополучия [5].

На рисунке 5 приведены экранные формы приложения «CareClinic», на которой видны все основные вкладки и функции приложения.

   
Рисунок 5 – Экранные формы приложения «CareClinic»



Достоинства системы:

* возможность ведения болезни;
* возможность отслеживания сна;
* возможности отслеживания симптомов;
* возможность отслеживания окружающей среды;
* возможность отключить ненужные функции.

Недостатки системы:

* бесплатно только пробный период (ежемесячная подписка);
* сильно различающиеся версии для разных платформ;
* отсутствует локализация;
* большая часть информации добавляется вручную;
* много недоработок.
  + 1. «Pills Time»

«Pills Time» – понятная напоминалка, помогающая заботиться о своем здоровье, не забыть принять вовремя таблетки или витамины. «Pills Time» - карманная аптека и удобное расписание приема медикаментов в одном приложении [6].

На рисунке 6 приведены экранные формы приложения «Pills Time», на которой видны все основные вкладки и функции приложения.

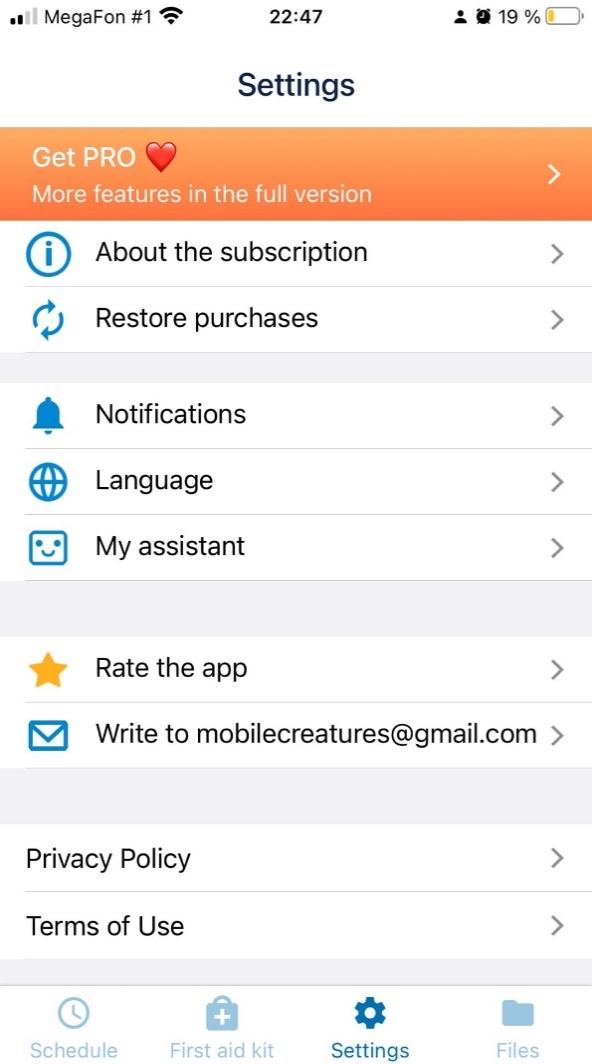
Достоинства системы:

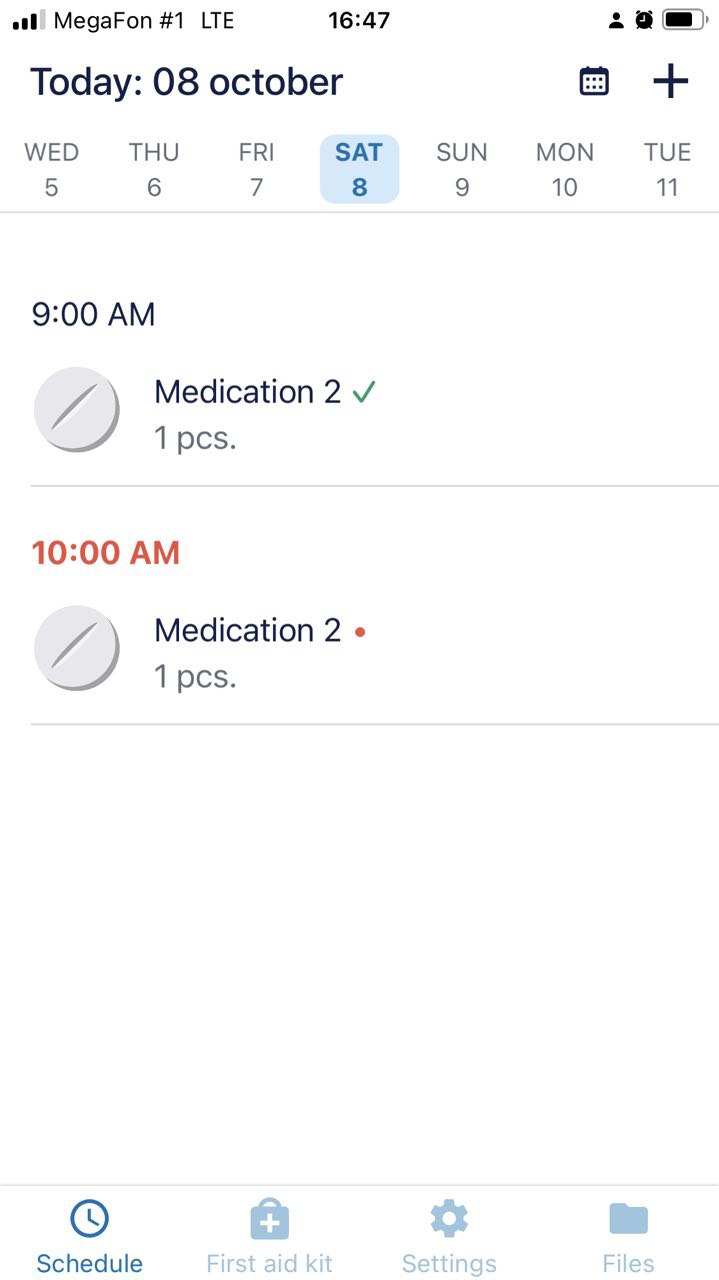
* удобный интерфейс;
* большое количество вариантов периодичности приема
* возможность отслеживать приемы пищи;
* возможность отслеживания посещений врачей.

Недостатки системы:

* нет библиотеки лекарств;
* вся информация вводится вручную;
* бесплатно доступен ограниченный набор функций.

На основании анализа возможностей систем-аналогов были сформулированы требования к разрабатываемой системе (см. таблицу 1).

  
Рисунок 6 – Экранные формы приложения «Pills Time»



* 1. Постановка задачи

Во время выполнения лабораторного практикума необходимо разработать мобильное приложение с функциями напоминания о необходимости приема лекарств, с помощью которого можно создавать свой список лекарств и настраивать время, дозировку и периодичность приема лекарств.

Перед тем как начать пользоваться функциями приложения, каждый пользователь должен пройти авторизацию, а именно: ввести номер телефона (в формате +7(YYY)XXX-XX-XX) и подтверждающий код, полученный по SMS, где YYY – код страны. При последующих входах в приложение будет использоваться механизм быстрой авторизации через touchId/faceId.

Таблица 1 – Сравнительные характеристики систем-аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Название системы  Название показателя | «Pills Time» | «CareClinic» | Разрабатываемое приложение |
| Локализация | да | нет | да |
| Библиотека лекарств | нет | да | да |
| Поиск лекарств в интернет-магазинах | нет | нет | да |
| Авторизация | нет | да | да |
| Выбор периодичности приема препаратов | да | да | да |
| Просмотр описания препарата | нет | нет | да |
| Ограничения на количество приемов одного препарата в день | Да | да | да |
| Версии для различных мобильных платформ | да | нет | да |
| Визуализация запланированных приемов лекарств в виде «календаря» | да | нет | да |

После авторизации пользователь может, перейдя на станицу добавления лекарственных препаратов, добавить необходимые ему лекарства в свою библиотеку, для поиска должна быть предусмотрена система фильтров по заранее заданным критериям: наличие сертификата, название препарата, компонент состава, страна производитель.

Для каждого добавленного лекарства пользователь сможет создавать напоминания о приеме с указанием дозировки, временем приема и периодичностью (каждый день, в определенный день недели).

Сервисная служба должна будет периодически проверять срабатывание созданных пользователями напоминаний и отправлять в приложение соответствующие уведомления.

После первой авторизации в приложении пользователю должен быть доступен его личный кабинет, где он может изменять свои персональные данные, такие как: имя, e-mail, а также подписываться/отписываться от оповещений.

Также пользователь по каждому отдельному препарату должен иметь возможность перейти на страницу с детальным описанием, включающим: сертификат, состав, формы выпуска, а также ссылки для перехода в интернет-магазин для последующего приобретения.

Для более удобного восприятия запланированных приемов лекарств, пользователю будет доступен «календарь» – специальная форма отображения напоминаний, в которой при выборе даты отображаются все препараты, которые ему необходимо принять в этот день.

Учетные данные пользователя, списки лекарственных препаратов, все справочник, агрегированная информация о напоминаниях будут храниться в базе данных на сервере.

Вся информация необходимая для работы системы будет храниться в базе данных, а именно:

* данные о пользователях (телефон, код, e-mail, имя, список лекарств, список напоминаний);
* данные о лекарственных препаратах (название, сертификат, дозировка, форма выпуска, состав, страна производитель, изображение);
* напоминания (лекарство, дата, время, дозировка, время до приема);
* сертификаты (номер, дата регистрации, статус);
* страны производители (название, изображение флага);

Также в системе должна быть организована выдача справочной информации о возможностях системы и о разработчиках.

Таким образом, системы должна решать следующие задачи:

Функции, реализуемые системой:

1) функции системы:

* + аутентификация пользователя в системе;
  + регистрация пользователя в системе;
  + рассылка напоминаний о приеме лекарств;
  + ускоренная авторизация;
  + отслеживание ранее отправленных уведомлений;
  + фильтрация лекарств по параметрам (название, состав, страна производитель, наличие сертификата);
  + автоматическое составление уведомлений для последующей отправки;
  + визуализация процессов работы с приложением;
  + проверка дублирования лекарств;
  + выдача справочной информации о системе;
  + выдача справочной информации о лекарственном препарате;
  + генерирование ссылок для поиска препарата в аптечных сетях.

1. функции пользователя:
   * регистрация в приложении (ввод телефона);
   * авторизация в приложении (ввод кода, отправленного по SMS);
   * возможность при повторной авторизации использовать touchId/faceId;
   * возможность по ссылке перейти в интернет магазин аптечных сетей;
   * возможность изменения своих персональных данных в личном кабинете (имя, e-mail);
   * возможность удалить свой аккаунт;
   * составление фильтров для поиска лекарств;
   * просмотр запланированных приемов;
   * отслеживание плана приема лекарства;
   * подтверждение приема лекарства;
   * добавление лекарств в свою библиотеку;
   * составление графика приема лекарств;
   * включение/отключение оповещений системы;
   * просмотр справочной информации.
2. Проектирование системы
   1. Выбор и обоснование архитектуры системы

Для разрабатываемого положения была выбрана клиент-серверная архитектура.

Клиент-сервер – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически [клиент](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) и [сервер](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) – это [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Обычно эти программы расположены на разных [вычислительных машинах](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) и взаимодействуют между собой через [вычислительную сеть](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) посредством [сетевых протоколов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) [7].

Серверная часть приложения будет реализована в виде web-приложения по модели REST API с поддержкой спецификации Open API в виде реализации swagger на платформе. Общение с клиентом (мобильное приложение) осуществляется посредством запросов/ответов по протоколу HTTP.

Application Program Interface (API) – правила, которым необходимо следовать для связи с другими программными системами. Разработчики внедряют или создают API-интерфейсы, чтобы другие приложения могли программно взаимодействовать с их приложениями [8].

Representational State Transfer (REST) – это программная архитектура, которая определяет условия работы API. Первоначально REST создавалась как руководство для управления взаимодействиями в сложной сети, такой как Интернет. Архитектуру на основе REST можно использовать для поддержки высокопроизводительной и надежной связи в требуемом масштабе. Ее можно легко внедрять и модифицировать, обеспечивая прозрачность и кроссплатформенную переносимость любой системы API [8].

RESTful API – это интерфейс, используемый двумя компьютерными системами для безопасного обмена информацией через Интернет.

Большинство бизнес-приложений должны взаимодействовать с другими внутренними и сторонними приложениями для выполнения различных задач. RESTful API поддерживают такой обмен информацией, поскольку они следуют безопасным, надежным и эффективным стандартам программного взаимодействия [8].

Спецификация OpenAPI – это формат описания API для REST API. Файл OpenAPI позволяет вам описать весь ваш API [9], включая:

* доступные конечные точки и операции на каждой конечной точке;
* ввод и вывод параметров работы для каждой операции;
* методы аутентификации;
* контактная информация, лицензия, условия использования и другая информация.

Спецификации API могут быть написаны на языке YAML или JSON. Формат прост в освоении и удобен как для людей, так и для машин.

Swagger – это набор инструментов с открытым исходным кодом, созданных на основе спецификации OpenAPI, которые могут помочь вам разрабатывать, создавать, документировать и использовать REST API [9]. Основные инструменты Swagger включают в себя:

* редактор Swagger – редактор на основе браузера, в котором можно писать определения OpenAPI;
* пользовательский интерфейс Swagger – отображает определения OpenAPI в виде интерактивной документации;
* swagger Codegen – генерирует заглушки сервера и клиентские библиотеки из определения OpenAPI;
* swagger Editor Next (бета–версия) - редактор на основе браузера, в котором вы можете писать и просматривать определения OpenAPI и AsyncAPI;
* swagger Core – библиотеки, связанные с Java, для создания, использования и работы с определениями OpenAPI;
* анализатор Swagger – автономная библиотека для анализа определений OpenAPI;
* swagger APIDom – предоставляет единую унифицированную структуру для описания API-интерфейсов на различных языках описания и форматах сериализации.

HyperText Transfer Protocol (HTTP) — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных [10].

Основой HTTP является технология «клиент-сервер», то есть предполагается существование:

* потребителей (клиентов), которые инициируют соединение и посылают запрос;
* поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

В качества служб push уведомлений будут использоваться соответствующие системные службы IOS/Android.

В качестве поставщика для СМС рассылок будет использоваться сервис «SMSAero».

Клиентская часть будет реализована в виде мобильного приложения для платформ под управлением IOS/Android.

* 1. Структурная схема системы

В основе структурного подхода к разработке ПС лежит алгоритмическая декомпозиция, когда система разбивается на функциональные подсистемы, которые, в свою очередь, делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур (алгоритмов). При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны. Разработка системы стала вестись по принципу «сверху-вниз», в отличие от применяемой ранее «снизу-вверх», когда при переходе от отдельных задач ко всей системе терялась целостность и возникали проблемы при информационной стыковке отдельных компонентов [11].

Структурная схема представляется как совокупность элементарных звеньев объекта и связей между ними, один из видов графической модели. Под элементарным звеном понимают часть объекта или системы управления, которая реализует элементарную функцию.

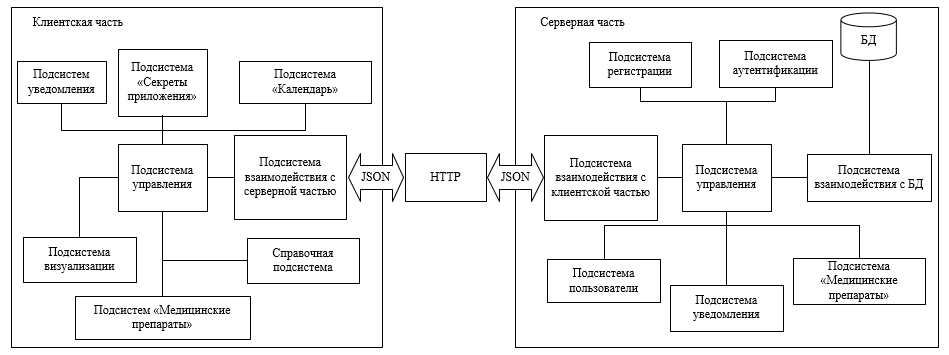
На рисунке 7 приведена структурная схема разрабатываемой системы.

В состав клиентской части системы входят следующие подсистемы:

1. подсистема взаимодействия с серверной частью, которая отвечает за получение/отправку данных на сервер;
2. подсистема управления, которая отвечает за связь всех подсистем на уровне приложения;
3. подсистема «Секреты приложения», которая отвечает за локальное хранилище секретов (в том числе личных данных) на устройстве;
4. подсистема уведомлений, которая отвечает за получение и обработку уведомлений;
5. подсистема «Календарь», которая отвечает за работу с календарем;
6. подсистема «Медицинские препараты», которая отвечает за работу с медицинскими препаратами;
7. подсистема визуализации, которая отвечает за смену визуальных состояний приложения;

В состав серверной части системы входят следующие подсистемы:

1. подсистема взаимодействия с клиентской частью, которая отвечает за получение/отправку данных клиенту;
2. подсистема управления, которая отвечает за связь всех подсистем на уровне приложения;

  
Рисунок 7 – Структурная схема системы

1. подсистема «Пользователи», которая отвечает за работу с пользователями;
2. подсистем уведомлений, которая отвечает за отправку уведомлений пользователям;
3. подсистема регистрации, которая отвечает за регистрацию пользователя;
4. подсистема аутентификация, которая отвечает за проверку прав доступ у пользователей;
5. подсистема «Медицинские препараты», которая отвечает за работу с медицинскими препаратами;
6. подсистема взаимодействия с БД, которая отвечает за отправку запросов в БД.
   1. Разработка прототипа интерфейса пользователя системы

Пользовательский интерфейс – совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером [12].

Разработка пользовательского интерфейса включает те же основные этапы, что и разработка программного обеспечения

* постановка задачи – определение типа интерфейса и общих требований к нему;
* анализ требований и определение спецификаций – определение сценариев использования и пользовательской модели интерфейса;
* проектирование – проектирование диалогов и их реализация в виде процессов ввода-вывода;
* реализация – программирование и тестирование интерфейсных процессов.

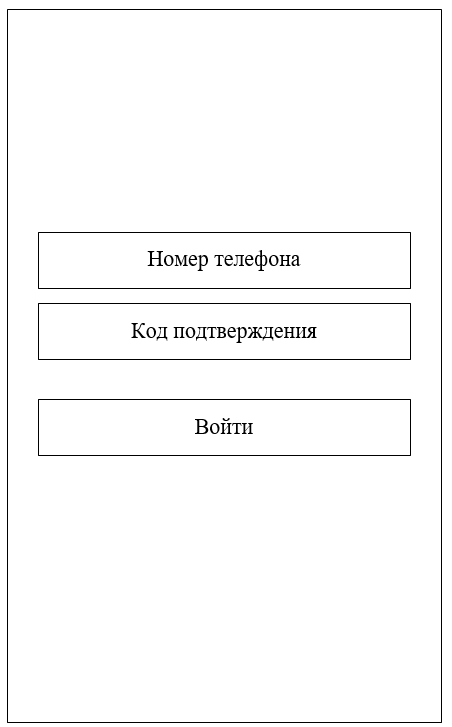
При проектировании пользовательских интерфейсов необходимо учитывать психофизические особенности человека, связанные с восприятием, запоминанием и обработкой информации.

Прототип ПО– это частичная или возможная реализация предлагаемого нового продукта. Прототипы позволяют решать 3 основные задачи [12]:

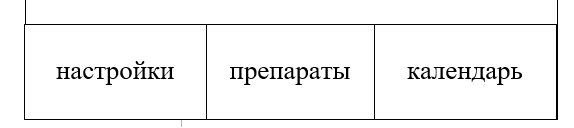
* прояснение и завершение процесса формулировки требований;
* исследование альтернативных решений;
* создание конечного продукта.

Основная цель создания прототипа – устранение неясностей на ранних стадиях процесса разработки.

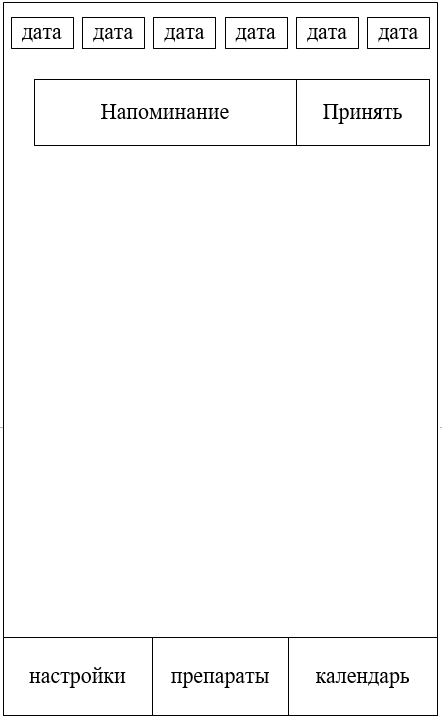
На рисунке 8 изображен прототип экрана авторизации пользователя. Для авторизации пользователю необходимо ввести номер своего сотового телефона в соответствующее поле, нажать кнопку “Получить код”, дождаться получения кода подтверждения по СМС, вести его во 2-е поле и нажать кнопку “Проверить код”. Последующие авторизации для упрощения механизма будут доступны через touchId/faceId.

  
Рисунок 8 – Прототип экрана авторизации

Каждая основной экран в нижней части будет иметь область быстрой навигации (рисунок 9). Каждая кнопка будет перенаправлять пользователя на соответствующий экран.

  
Рисунок 9 – Прототип экранной формы поиска лекарств

После авторизации откроется основной экран приложения – «Календарь» (рисунке 10). Данный экран предназначена для быстрой навигации по прошедшим и предстоящим напоминаниям в разрезе дня. Пользователь сможет выбрать любой день из списка и увидеть все напоминания в этот день, а также подтвердить, что принял лекарство.

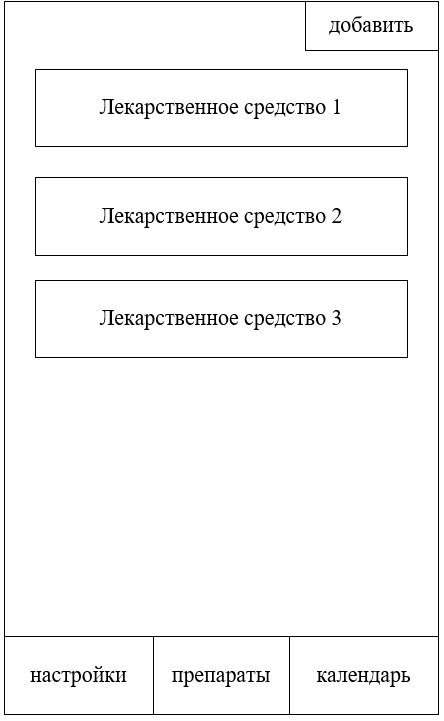
  
Рисунок 10 – Прототип экрана «Календарь»

На рисунке 11 изображен прототип экрана библиотеки лекарств пользователя. На этой странице будут отображены все лекарственные препараты, которые пользователь добавил в свою библиотеку. Для каждого препарата из списка будет доступен перечень действий: показать детализацию, настроить уведомления и удалить из списка. Все эти действия вызываются из контекстного меню.

Для добавления нового препарата в список пользователь должен нажать на кнопку “Добавить”, после чего откроется экран поиска лекарств.

Для настройки напоминаний пользователь должен будет нажать соответствующую кнопку в контекстном меню, после чего откроется экран настройки напоминаний.

Для просмотра описания лекарственного препарата пользователь должен будет нажать на соответствующую кнопку в контекстном меню, после чего откроется соответствующий экран с детализации по препарату.

  
Рисунок 11 – Прототип экрана библиотека лекарств пользователя

На рисунке 12 изображен прототип экрана поиска лекарственных средств. Пользователю должны быть доступны на выбор все лекарственные препараты, добавленные в систему. Для более удобного поиска должна быть предусмотрена система фильтров. В фильтре у пользователя должна быть возможность выбрать страну изготовителя, указать название препарата, указать элемент состава и отфильтровать по наличию сертификата.

Для добавления препарата в свою библиотеку, пользователь должен будет нажать советующую кнопку в контекстном меню. После чего пользователя автоматически перенаправит на экран со список его лекарственных препаратов.

Для просмотра описания лекарственного препарата пользователь должен будет нажать на соответствующую кнопку, после чего откроется соответствующий экран с детализации по препарату.

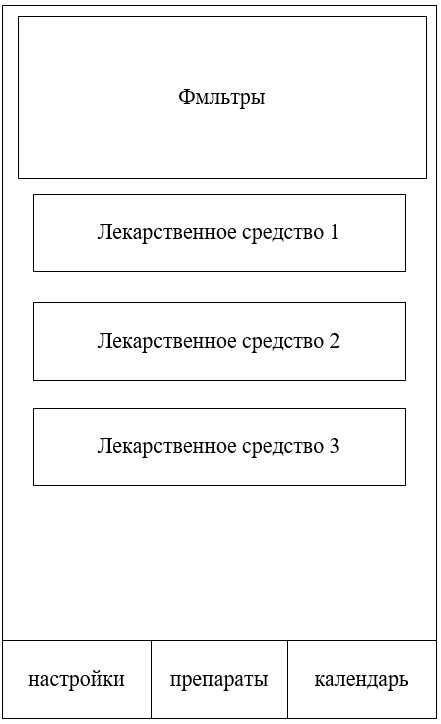
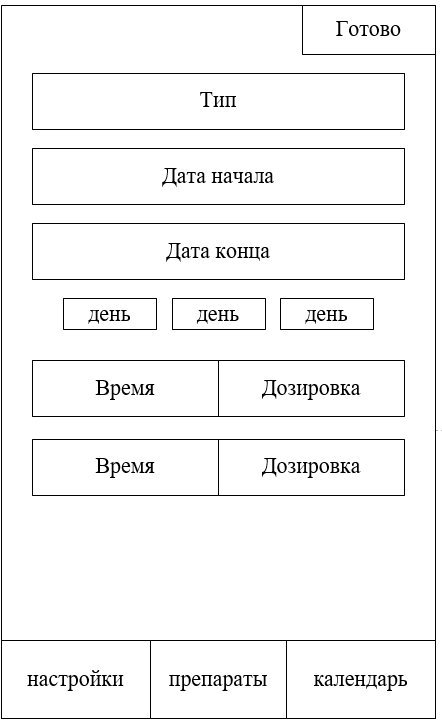


Рисунок 12 – Прототип экрана поиска лекарств

На рисунке 13 изображен прототип экрана создания напоминаний. На этом экране пользователю должен быть доступен функционал по созданию поминаний для конкретного препарата.

  
Рисунок 13 – Прототип экрана создания напоминаний

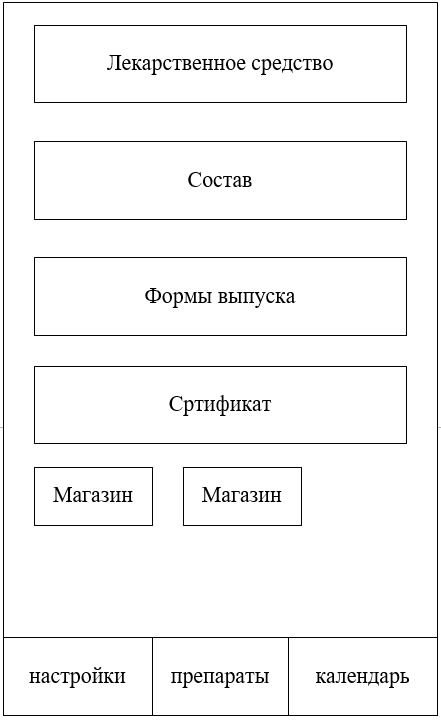
У пользователя должна быть возможность выбрать тип периодичности, дату начала и конца приема. В зависимости от выбранного типа периодичности должны появляться дополнительные элементы управления.

Далее пользователю должен будет добавить необходимое ему кол-во напоминании в день, указать для них время приема и дозировку препарата.

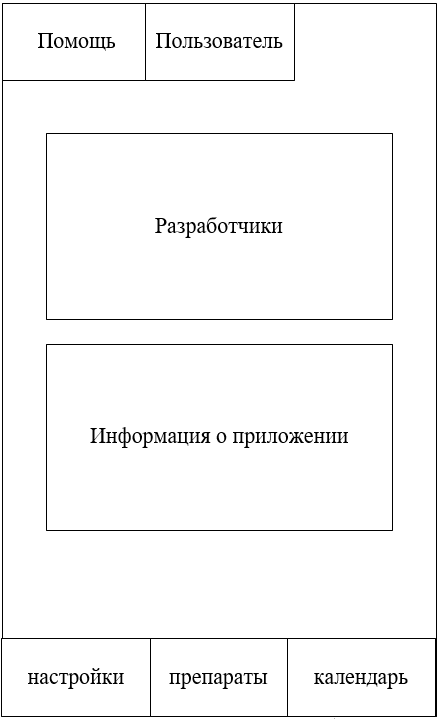
Как только пользователь закончит работу с напоминаниями, он должен будет нажать кнопку “Готово”, после чего его автоматически перенаправит на экран со список его лекарственных препаратов.

На этом же экране у пользователя должна быть возможность в любой момент вносить изменения в ранее созданные напоминания.

На рисунке 14 изображен прототип экрана просмотра детализации по лекарству. На данном экране пользователь должен иметь возможность просмотреть всю имеющуюся информацию по препарату, такую как: название, состав, формы выпуска, сертификат, прочие характеристики. А также по соответствующим кнопкам перейти на интернет магазины, соответствующих аптек.

  
Рисунок 14 – Прототип экрана детализации по лекарству

На рисунке 15 изображен прототип экрана «Помощь». На данном экране должна будет отображаться общая информация о приложении, а также информация о разработчиках.

  
Рисунок 15 – Прототип экрана «помощь»

На рисунке 16 изображен прототип экрана личного кабинета пользователя. На данном экране у пользователя должна быть возможность вносить изменена в свои личные данные путем изменения соответствующих полей, а также включать или отключать подписки на оповещения.

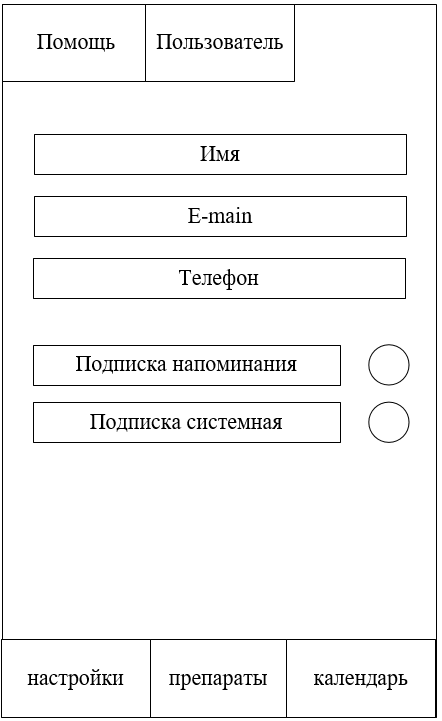


Рисунок 16 – Прототип экрана личного кабинета пользователя

На рисунке 17 приведена навигационная модель разрабатываемого приложения.

* 1. Разработка информационно-логического проекта системы
     1. Язык UML

Для специфицирования (построения точных, недвусмысленных и полных моделей) системы и ее документирования используется унифицированный язык моделирования UML.

Язык UML – язык широкого профиля и представляет собой открытый стандарт, в котором используются различные графические обозначения, для создания абстрактной модели системы. Этот язык создавался для того, чтобы

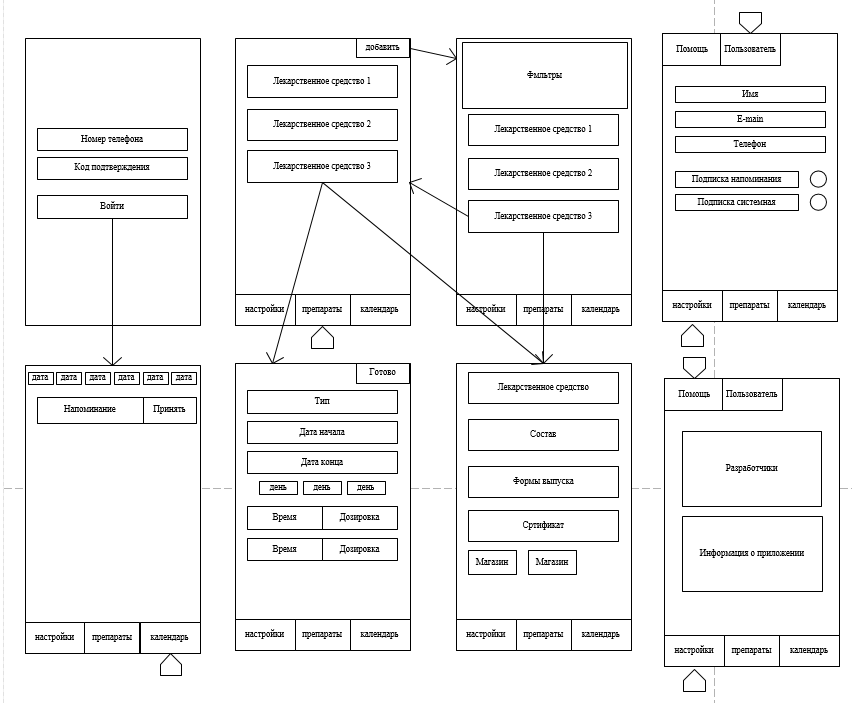
****

Рисунок 17 ‒ Навигационная модель приложения



обеспечить определение, визуализацию, документирование и проектирование всевозможных программных систем. Важно знать, что сама по себе UML-диаграмма не представляет собой язык программирования, но при этом предусматривается возможность генерации на ее основе отдельного кода на других языках программирования [13].

Применение UML не заключается только в моделировании всевозможного ПО. Также данный язык активно используется для моделирования различных бизнес-процессов, ведения системного проектирования, а также отображения организационных структур. С помощью UML можно обеспечить полное соглашение в используемых графических обозначениях, чтобы представить общие понятия: компонент, обобщение, класс, поведение и агрегация. За счет этого достигается высокая степень концентрации на архитектуре и проектировании.

Использование UML обеспечивает следующие преимущества:

* UML – объектно-ориентированный язык, поэтому методы описания результатов анализа и проектирования очень близки к методам программирования на современных объектно-ориентированных языках;
* UML позволяет описывать систему почти со всех возможных точек зрения и различные аспекты поведения системы;
* диаграммы UML относительно просты для чтения после достаточно быстрого ознакомления с синтаксисом;
* UML – расширяемый язык и позволяет вводить собственные текстовые и графические стереотипы, позволяющие применять ᴇᴦο не только в сфере программной инженерии.
  + 1. Диаграмма вариантов использования

Диаграмма вариантов использования представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм. На ней изображаются отношения между актерами и вариантами использования [13].

Диаграмма вариантов использования отображает на себе все отношения, которые возникают между актерами, а также различными вариантами использования и является исходным концептом системы в процессе проектирования и разработки.

Данная диаграмма включает: актеров, варианты использования, отношения между ними.

На рисунке 18 приведена диаграмма вариантов использования со стороны пользователя.

На рисунке 19 приведена диаграмма вариантов использования со стороны системы.

Основными функциями пользователя являются:

* пройти регистрацию – регистрация пользователя происходит по номеру телефона во время первой авторизации;
* пройти авторизацию – авторизация происходит путем ввода номера телефона, на который будет отправлен авторотационный код и последующей отправки полученного кода для сравнения. После первой авторизации будет доступен ускоренный вариант по touchId/faceId;
* удалить аккаунт – пользователь может в любой момент отчисть все свои данные, удалив свой аккаунт, через личный кабинет;
* создать напоминание – создание напоминаний для конкретного препарата доступно на специальном экране, на котором пользователь указывает даты начала и конца приема, тип периодичности и дозировку;
* изменить подписки на оповещения – через личный кабинет пользователь может включать или отключать подписки на оповещения (системные, о приеме лекарств);
* изменить личные данные – через личный кабинет пользователь может изменять свои личные данные (имя, e-mail);
* посмотреть справочную информацию – пользователь может посмотреть справочную информацию как о самой системе, так и о разработчиках;
* добавить препарат в библиотеку – добавление препаратов в библиотеку происходит на отдельном экране. Для удобства предусмотрена система фильтров, которая позволяет указать такие параметры как страну производителя, элемент состава, название препарата или наличие сертификата;
* перейти в интернет магазин для поиска препарата – на экране детализации по препарату доступны ссылки на самые популярны интернет площадки продажи лекарственных препаратов;
* удалить препарат из библиотеки – пользователь может в любой момент удалить ранее добавленные препараты, из своей библиотеки на специальном экране;
* посмотреть детализацию по препарату – пользователь может посмотреть всю имеющуюся информацию по лекарственному средству на отдельном экране;
* посмотреть все напоминания – пользователь на специальном экране может просмотреть все прошедшие и предстоящие приемы лекарства, а также подтвердить их прием;

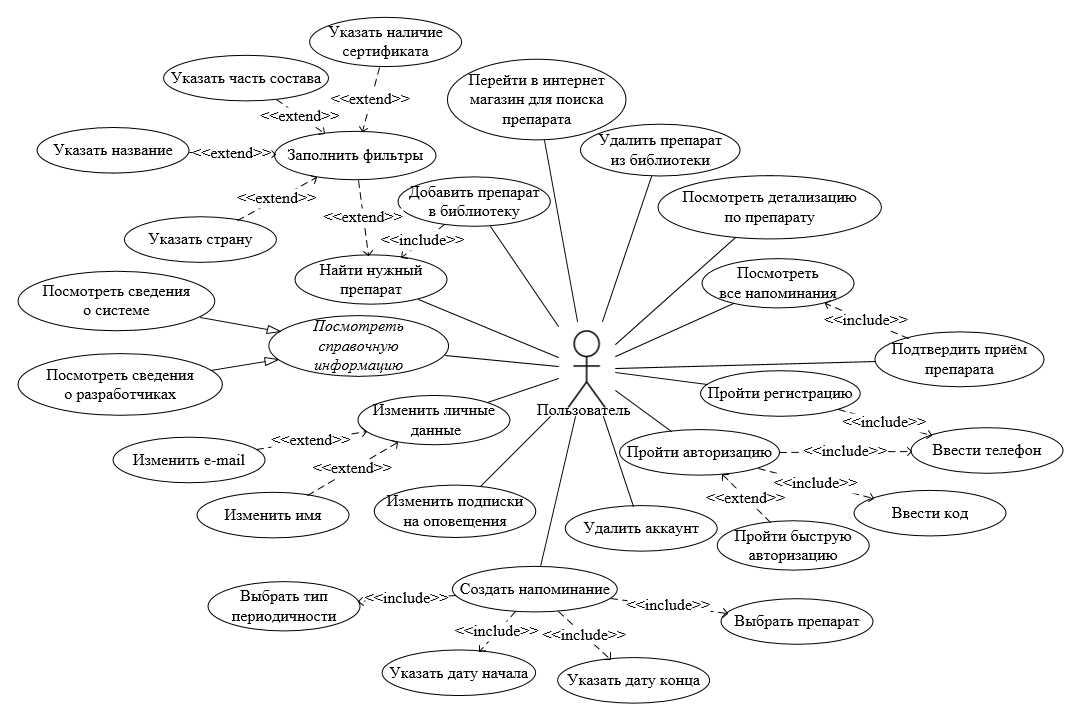
Основными функциями системы являются:

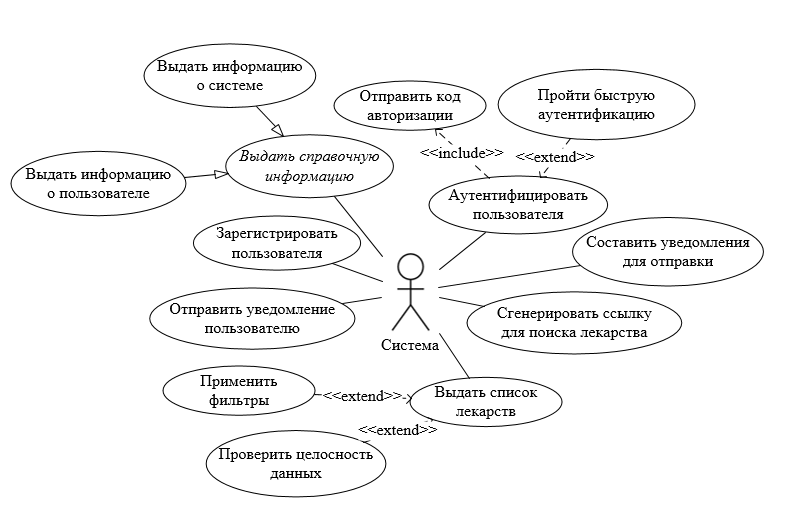
* аутентифицировать пользователя – аутентификация пользователя происходит путем сравнения сгенерированного кода и отправленным пользователем во время авторизации;
* зарегистрировать пользователя – регистрация пользователя происходит по номеру телефона во время первой авторизации;
* сгенерировать ссылку для поиска лекарства – для каждого лекарственного средства, добавленного в систему, генерируется ссылка для поиска его в наиболее популярных интернет магазинах;
* выдать список лекарств – система, по запросу пользователя, выдает список лекарств, добавленных в системы, с учетом фильтров;
* отправить уведомление пользователю – система отправляет push уведомления пользователю в различные, заранее определенные, топики (системные сообщения, уведомления о приеме лекарств);
* выдать справочную информацию – система может выдать справочную информацию как о пользователе, работающем с системой, так и о самой системе.
  + 1. Диаграмма классов

Диаграммы классов – это наиболее часто используемый тип диаграмм, которые создаются при моделировании объектно-ориентированных систем, они показывают набор классов, интерфейсов и коопераций, а также их связи. На практике диаграммы классов применяют для моделирования статического представления системы, они служат основой для целой группы взаимосвязанных диаграмм – диаграмм компонентов и диаграмм размещения [14].

На рисунке 20 приведена диаграмма классов.

В таблице 2 приведено описание классов.

  
Рисунок 18 − Диаграмма вариантов использования системы (пользователь)

  
Рисунок 19 − Диаграмма вариантов использования системы (система)

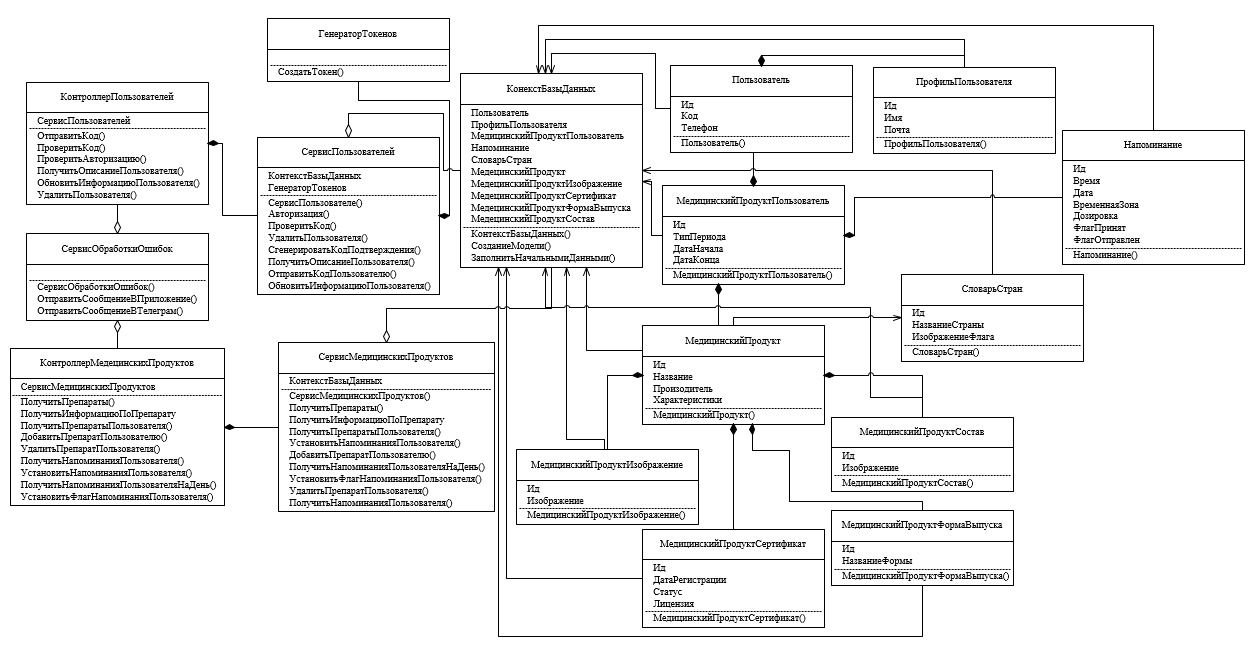
Рисунок 20 − Диаграмма классов системы

Таблица 2 – Описание классов системы

|  |  |
| --- | --- |
| Название класса | Назначение |
| Пользователь | Класс, хранящий в себе всю информацию о пользователе. |
| ПрофильПользователя | Класс, хранящий в себе всю личную информацию о пользователе. |
| Напоминание | Класс, хранящий в себе всю информацию о напоминаниях. |
| МедицинскийПродукт | Класс, хранящий в себе всю информацию о медицинских продуктах. |
| МедицинскийПродуктСертификат | Класс, хранящий в себе всю информацию о сертификатах медицинских продуктов. |
| СловарьСтран | Класс, хранящий в себе всю информацию о странах. |
| МедицинскийПродуктСостав | Класс, хранящий в себе всю информацию о составе медицинских препаратов. |
| МедицинскийПродуктФорма Выпуска | Класс, хранящий в себе всю информацию о формах выпуска медицинского препарата |
| МедицинскийПродуктИзображение | Класс, хранящий в себе всю информацию о изображениях для медицинских препаратов. |
| МедицинскийПродуктПользователь | Класс, хранящий в себе всю информацию о пользовательской библиотеке препаратов. |
| КонекстБазыДанных | Класс, отвечающий за подключение к БД. |

Продолжение таблицы 2

|  |  |
| --- | --- |
| 1 | 2 |
| ГенераторТокенов | Класс, отвечающий за генерацию токенов пользователей. |
| СервисПользователей | Класс, отвечающий за логику работу с пользователями. |
| СервисМедицинскихПродуктов | Класс, отвечающий за логику работу с медицинскими продуктами. |
| КонтроллерПользователей | Класс, обрабатывающий запросы по работе с пользователями. |
| СервисОбработкиОшибок | Класс, отвечающий за логику обработки ошибок приложения. |
| КонтроллерМедецинскихПродуктов | Класс, обрабатывающий запросы по работе с медицинскими препаратами. |

* 1. Логическая модель данных

Проектирование БД является одной из важнейших составных частей процесса создания системы. База данных, рассматриваемая как сложная система, разрабатывается с использованием тех же принципов, что и система в целом. При проектировании баз данных обычно выделяют три уровня абстракции, на которых происходит последовательное уточнение модели: концептуальный (семантический уровень представления данных в виде абстрактных понятий, учитывающих особенности предметной области), логический (уровень представления в виде структуры данных – сущностей, атрибутов и связей) и физический (уровень реализации базы данных) [15].

Логическая информационная модель – модель данных, в которой учитывается способ логического хранения данных в памяти ЭВМ.

На рисунке 21 приведена логическая модель данных системы. В таблицах 3-12 приведено описание сущностей БД.

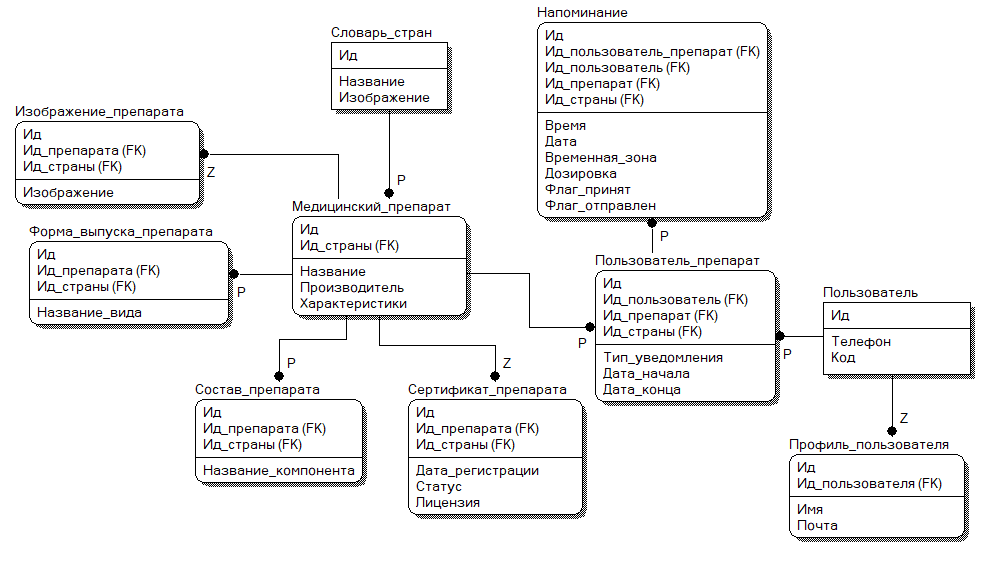
  
Рисунок 21 – Логическая модель данных системы

Таблица 3 – Сущность «Напоминание»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид | Целый | Уникальный идентификатор |
| Время | Символьный[11] | Время в которое нужно отправить уведомление |
| Дата | Дата | Дата в которую нужно отправить уведомление |
| Временная зона | Время | Временная зона пользователя для преобразования времени |
| Дозировка | Целый | Дозировка препарата, которую нужно принять |
| Флаг принят | Логический | Флаг принял ли пользователь препарат |
| Флаг отправлен | Логический | Флаг отправлено ли напоминание пользователю |

Таблица 4 – Сущность «Пользователь»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид | Целый | Уникальный идентификатор |
| Телефон | Символьный[11] | Телефон пользователя, используемый для авторизации |
| Код подтверждения | Символьный[4] | Сгенерированный код для авторизации |

Таблица 5 – Сущность «Профиль пользователя»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид | Целый | Уникальный идентификатор |
| Имя | Символьный[50] | Имя пользователя |
| Почта | Символьный[50] | Почта пользователя |

Таблица 6 – Сущность «Словарь стран»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид | Целый | Уникальный идентификатор |
| Название | Символьный[11] | Название страны |
| Изображение | Массив байт | Изображение флага страны |

Таблица 7 – Сущность «Состав препарата»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид | Целый | Уникальный идентификатор |
| Название компонента | Символьный[50] | Название компонента в составе препарата |

Таблица 8 – Сущность «Форма выпуска препарата»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид | Целый | Уникальный идентификатор |
| Название вида | Символьный[300] | Вид формы выпуска препарата |

Таблица 9 – Сущность «Сертификат препарата»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид | Целый | Уникальный идентификатор |
| Дата регистрации | Дата | Дата регистрации препарата |
| Статус | Символьный[11] | Статус зарегистрированного препарата |
| Лицензия | Символьный[20] | Номер лицензии препарата |

Таблица 10 – Сущность «Медицинский препарат»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид | Целый | Уникальный идентификатор |
| Название | Символьный[50] | Название препарата |
| Производитель | Символьный[50] | Производитель препарата |
| Характеристики | Символьный[50] | Прочие характеристики препарата |

Таблица 11 – Сущность «Изображение препарата»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид | Целый | Уникальный идентификатор |
| Изображение | Массив байт | Изображение препарата |

Таблица 12 – Сущность «Пользователь-Препарат»

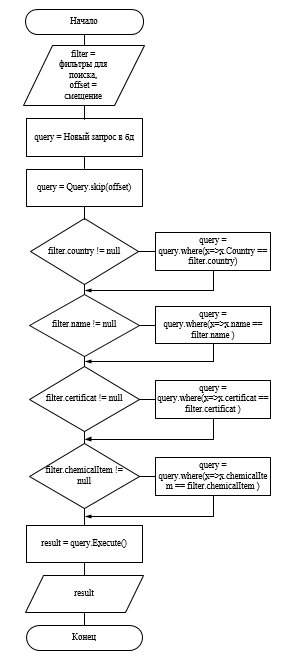
|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид | Целый | Уникальный идентификатор |
| Тип уведомления | Целый | Тип периодичности приема препарата |
| Дата начала | Дата | Дата начала приема препарата |
| Дата конца | Дата | Дата конца приема препарата |

* 1. Выбор и обоснование алгоритмов обработки данных

Алгоритм — это некая последовательность действий (вычислительных шагов), благодаря чему происходит преобразование входных данных в выходные данные. Алгоритмы используются при обработке данных как в информатике и программировании [16]. Алгоритм как последовательность шагов позволяет решать конкретную задачу и должен:

* работать за конечный объем времени. Если алгоритм не способен разобраться с проблемой за конечное количество времени, можно сказать, что он бесполезен;
* иметь четко определенные инструкции, порядок. Любой шаг должен точно определяться. Его инструкции должны быть однозначны для любой последовательности шагов;
* быть пригодным к использованию;
* алгоритм должен быть способен решить проблему, для устранения которой его создавали.

На рисунке 22 приведена схема алгоритма наложения фильтров на запрос в базу данных для поиска лекарств.

  
Рисунок 22 – Схема алгоритма наложения фильтров на  
 запрос в базу данных для поиска лекарств

* 1. Выбор и обоснование комплекса программных средств
     1. Выбор языка программирования

В качестве языка программирования был выбран язык C#. Основными преимуществами данного языка являются [17]:

* + популярность языка;
  + объектная ориентированность;
  + поддержка инкапсуляции, наследования и полиморфизма;
  + бесплатный ряд инструментов для небольших компаний и некоторых индивидуальных разработчиков (Visual Studio, облако Azure, Windows Server и др.);
  + скорость разработки – С# позволяет стартовать разработку быстрее, а это позволяет быстрее получить прототип решения;
  + производительность и требовательность к ресурсам – используя C# проще написать код, удовлетворяющий критериям «простоты разработки» и «красоты кода» одновременно;
  + множество библиотек с .net идет в базе, плюс к ним множество свободно доступных библиотек, это покрывает практически все первостепенные задачи разработки под Windows;
  + удобство отладки.
    1. Выбор среды разработки

В качестве среды программирования была выбрана Microsoft Visual Studio.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры, и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения и веб-службы [18].

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения [18].

* + 1. Выбор системы управления базами данных

В качестве система управления реляционными базами данных была выбрана Microsoft SQL Server [19]:

Основные преимущества:

* + масштабируемость и надежность;
  + скорость создания решений в сочетан c .Net;
  + возможность обработки вычислений в оперативной памяти;
  + возможность взаимодействия с публичным облаком Azure;
  + улучшенное колоночное хранилище в памяти;
  + улучшенная технология AlwaysOn;
  + высокая производительность;
  + отказоустойчивость;
  + поддержка локальной базы данных;
  + встроенные интеграции для работы с Visual Studio.
    1. Выбор платформы для создания мобильных приложений

В качестве платформы для разработки мобильных приложение выбран .NET MAUI.

NET Multi-Platform App UI (.NET MAUI) – это кроссплатформенная платформа для создания собственных мобильных и классических приложений с помощью C# и XAML [20].

.NET MAUI объединяет API Android, iOS, macOS и Windows в единый API, который позволяет выполнять однократную запись в любом месте разработки, обеспечивая дополнительный доступ к каждому аспекту каждой собственной платформы.

С помощью .NET MAUI можно разрабатывать приложения, которые могут работать в Android, iOS, macOS и Windows из одной общей базы кода.

Основные возможности:

* + один общий проект, предназначенный для Android, iOS, macOS и Windows;
  + упрощенный выбор целевого объекта отладки для запуска приложений MAUI .NET;
  + общие файлы ресурсов в рамках одного проекта;
  + отдельный манифест приложения, указывающий название, идентификатор и версию приложения;
  + доступ к API и средствам для конкретной платформы при необходимости;
  + одна кроссплатформенная точка входа приложения;
  + горячая перезагрузка.
    1. Выбор поддерживаемых платформ

Клиентская часть

Самыми популярными и по сути, единственными мобильными платформами на данный момент являются IOS и Android. Платформа .NET MAUI позволяет разрабатывать приложения, нацеленные на эти платформы. Поэтому в качестве поддерживаемых платформ будут выбраны IOS и Android.

Серверная часть

СУБД Microsoft SQL Server работает только на операционной системе под управлением Windows.

Платформа ASP .NET core является кроссплатформенной и позволяет создавать релизные сборки (готовые к публикации), нацеленные на разные операционные системы (Windows, все Linux подобные системы). Главным фактором является наличие установленного web сервера, с возможность работы как прокси сервер (IIS, Nginx и прочие).

* + 1. Выбор поставщика СМС-сообщений

В качестве поставщик СМС сообщений был выбран сервис «SMSAero». «SMSAero» - сторонний сервис, выступающий в роле поставщика СМС сообщений и предоставляющий услуг по массовым рассылкам СМС сообщений.

Преимущества:

* + относительно низкая цена 1 СМС;
  + полностью документированное API;
  + наличие личного кабинета;
  + знакомая и ранее используемая система.
    1. Выбор поставщика PUSH-уведомлений

По соглашениям Apple и Google поставщиками push-уведомлений для их платформ могут быть только их собственные службы, реализованные на самих платформах.

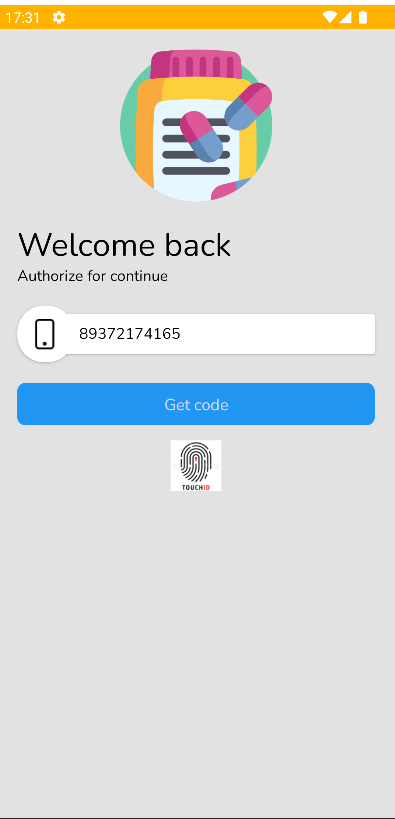
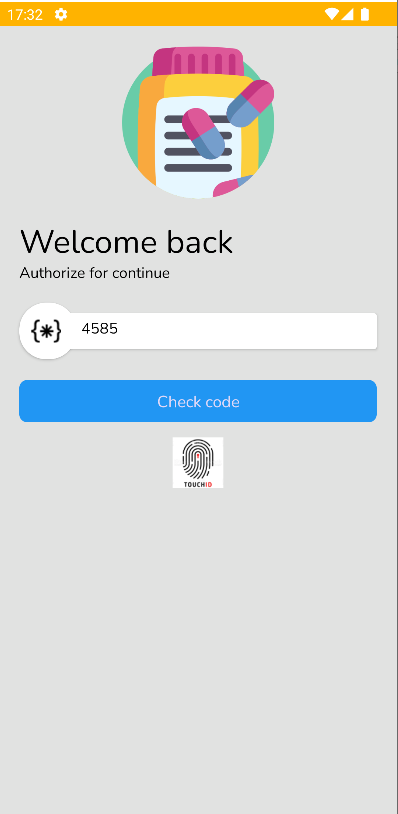
Для Android и IOS в качестве поставщика push уведомлении будет использоваться FCM т.к. это решение обладает встроенными интеграциями с сервисами Google и Apple.

Firebase Cloud Messaging (FCM) – это кроссплатформенное решение от Google для обмена сообщениями между устройствами.

1. Реализация системы
   1. Разработка и описание интерфейса пользователя

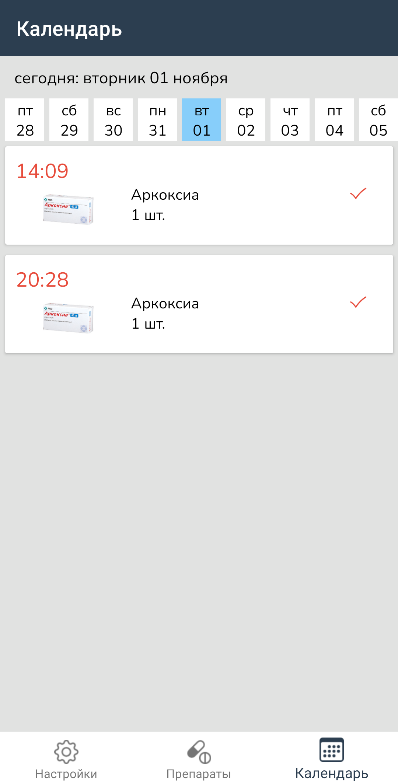
При разработке интерфейса пользователя необходимо учитывать, что интерфейс должен быть не интуитивно доступным и понятным пользователю, чтобы пользователь мог быстро разобраться с работой системы и легко взаимодействовать с ней.

При запуске приложения открывается экрн авторизации (рисунок 23). Регистрация происходит автоматически при первой авторизации пользователя.

   
Рисунок 23 – Экран авторизации

Чтобы авторизоваться в приложении, пользователь вводит номер телефона в специальное поле и должен надать на кнопку «Get code». После чего ожидает СМС и вводит полученный код в специальное поле и нажимает на кнопку «Check code».

После авторизации система открывает экран «Календарь» (рисунок 24). В верхней части экрана отображается список прошедших и будущих дат, при выборе даты из этого списка отображаются все напоминания за этот день. На этом же экране пользователь может подтвердить прием препарата, нажав на конкретное напоминание.

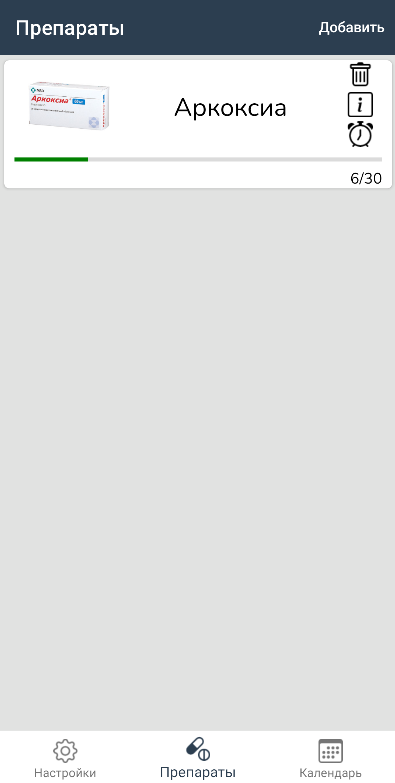
  
Рисунок 24 – Экран «Календарь»

Для навигации по приложению на каждом экране в нижней части присутствует навигационное меню (рисунок 25).

  
Рисунок 25 – Навигационное меню

Каждая кнопка навигации перенаправляет пользователя на соответствующий экран.

На рисунке 26 изображен экран «Препараты». На данном экране показан список всех лекарственных средств, добавленных пользователем в свою билиотеку.

  
Рисунок 26 – Экран «Препараты»

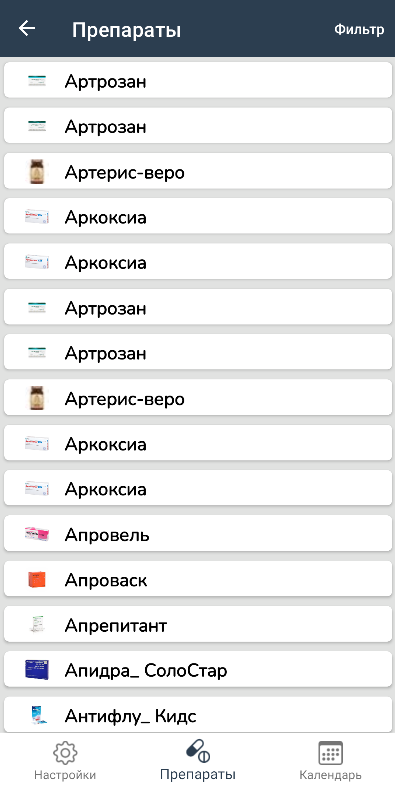
Для того чтобы удалить запись, необходимо нажать на кнопку с изображением .

Для того чтобы просмотреть детализацию по препарату, необходимо нажать на кнопку .

Для того чтобы настроить напоминания о приеме препарата необходимо нажать на кнопку .

Под каждым лекарственным препаратом отображается полоса прогресса по принятию лекарств.

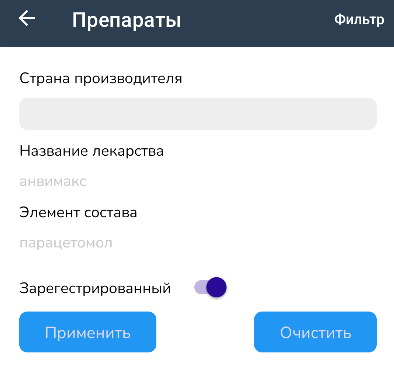
Для того чтобы добавить новый лекарственный препарат в свою библиотеку, необходимо нажать на кнопку «Добавить», после чего открывается экран поиска лекарственных средств (рисунок 27).

  
Рисунок 27 – Экран поиска лекарственных средств

На данном экране отображается динамический список всех лекарственных средств, который подгружается автоматически из БД.

Для добавления препарата в свою библиотеку пользователю нужно нажать на необходимый препарат и, удерживая нажатым, потянуть элемент списка вправо.

Для более удобного и быстрого поиска необходимого лекарственного препарата предусмотрена система фильтров для того, чтобы меню фильтров отобразилось нужно нажать на кнопку «Фильтр». После чего откроется специальная форма (рисунок 28). На данной форме можно указать страну производителя, название лекарства, элемент состава, а также установить флаг наличия сертификата у препарата.

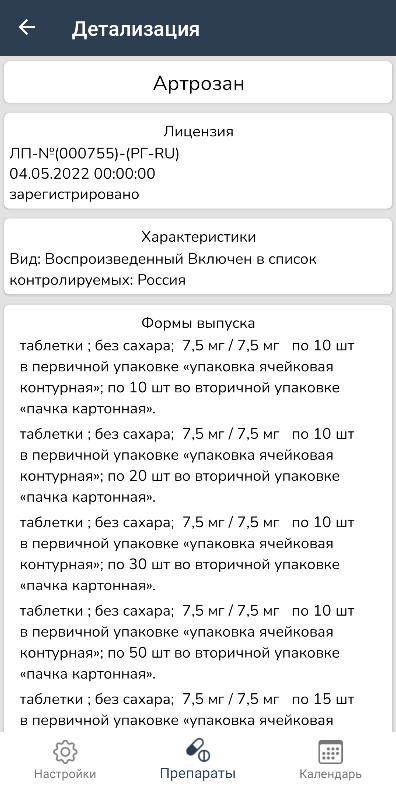
  
Рисунок 28 – Фильтр на экране поиска лекарственных средств

Чтобы применить фильтр, необходимо нажать на кнопку «Применить», а для того чтобы сбросить фильтры, на кнопку «Очистить».

На рисунке 29 изображен экран с детализацией лекарственного препарата. На данном экране показаны все имеющиеся в системе характеристики лекарственного препарата, а так же ссылки для перехода в интернет-магазины для покупки препарата.

На рисунке 30 изображен экран «Пользователь». На данном экране отображены личные данные пользователя с возможность их изменения.

Для изменения доступны поля «Email» и «Телефон». Чтобы применить изменения, нужно внести изменения в соответствующие поля и нажать на кнопку «Обновить аккаунт».

  
Рисунок 29 – Экран с детализацией лекарственного средства

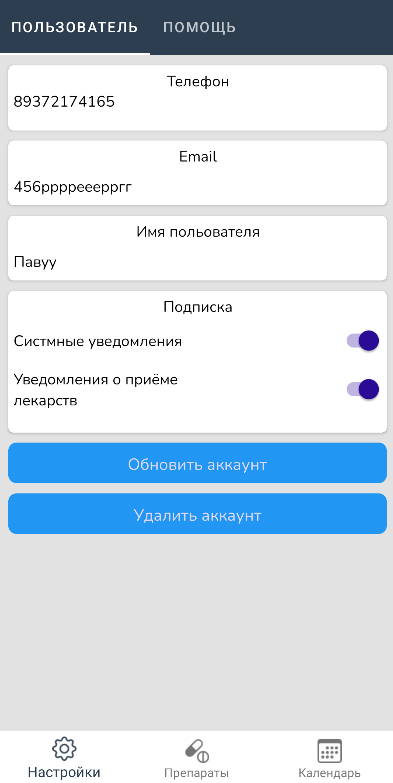
При нажатии на кнопку «Удалить аккаунт» произойдет полная очистка всех пользовательских данных.

Также на данном экране предоставлены функции отписки/подписки на оповещения посредством соответствующих переключателей.

На рисунке 31 изображен экран «Помощь». На данном экране отображена вся основная информация о приложении и разработчиках, а также присутствует индикатор наличия интернета.

Для того чтобы обновить статус индикатора наличия интернета необходимо нажать на кнопку «Проверить».

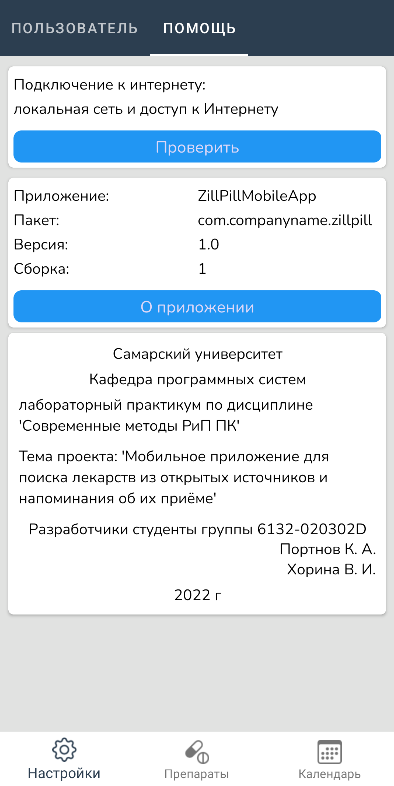
Для того чтобы посмотреть информацию о приложении, необходимо нажать на кнопку «О приложении».

  
Рисунок 30 – Экран «Пользователь»

На рисунке 32 изображен экран «Напоминания». На данном экране доступен функционал по созданию напоминаний для конкретного лекарственного препарата.

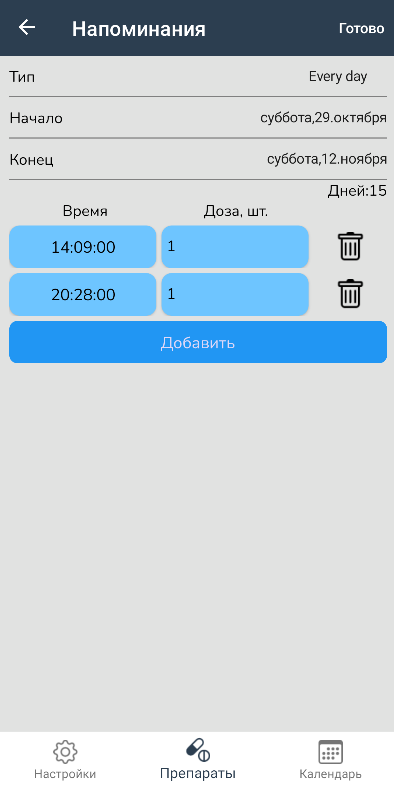
Для создания напоминаний имеется возможность выбрать тип периодичности (каждый день или в определенный день недели), дату начала и конца приема. В зависимости от выбранного типа периодичности изменяется отображаемая форма и добавляются дополнительные элементы управления.

Для добавления нового напоминания нужно нажать на кнопку «Добавить» и указать время приема и дозировку препарата. Чтобы удалить напоминание, нужно нажать на кнопку .

  
Рисунок 31 – Экран «Помощь»

Для завершения работы с напоминаниями и сохранения всех изменений, необходимо нажать на кнопку «Готово», после чего система открывает экран со список лекарственных препаратов пользователя.

На этом же экране у пользователя есть возможность в любой момент вносить изменения в ранее созданные напоминания, открывая данный экран для любого лекарственного средства из своего списка лекарств.

  
Рисунок 32 – Экран «Напоминания»

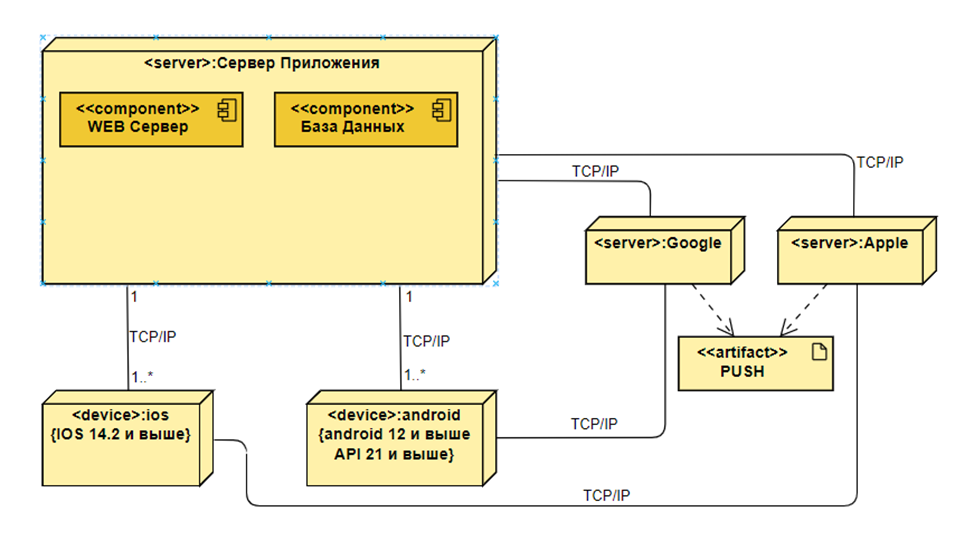
* 1. Диаграммы реализации

Диаграммы реализации предназначены для отображения состава компилируемых и выполняемых модулей системы, а также связей между ними. Диаграммы реализации разделяются на три конкретных вида: диаграмма классов, диаграмма компонентов (component diagrams) и диаграмма развертывания (deployment diagrams) [22].

* + 1. Диаграмма развертывания

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их [23].

На рисунке 33 приведена диаграмма развертывания системы.

  
Рисунок 33 – Диаграмма развертывания системы

Основные узлы:

* + устройства клиентов на платформе ios/android – мобильное устройство с установленным приложением;
  + сервер положения, на котором развернуты web сервер и сервер базы данных;
  + сервера ios/google для обработки push-уведомлений.
    1. Диаграмма классов

В соответствии со спецификацией, приведенной в п. 2.5.6, и с учетом выбранного языка программирования (см. п. 2.8.1) разработана диаграмма классов системы (этап реализации).

На рисунке 34 приведена диаграмма классов (этап реализации) инфраструктурного слоя. На данном слое отображены все основные сервисы, в которых заключена вся бизнес логика приложения и их связи между собой.

На рисунке 35 приведена диаграмма классов (этап реализации) доменного слоя. На данном слое отображены все основные классы сущностей и их связи между собой и контекстом базы данных.

* 1. Физическая модель данных

Физический уровень представления данных является самым низким уровнем, на котором определяются форматы хранимых данных, способ их размещения на носителях, выбор способа доступа к ним. Физическая модель БД, разработанной в системе, представлена на рисунке 36.

Подробное описание атрибутов сущностей физической модели данных приведено в таблицах 13-22.

Первичные ключи выделены жирным шрифтом, а внешние – курсивом.

Таблица 13 – Таблица «Словарь стран»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **id** | **Уникальный идентификатор** | **int** | **4** |
| name | Название страны | varchar(50) | 102 |
| imageData | Изображение флага в двоичном формате | varbinary(max) | 2147483648 (max)  1 550 (avg) |
| Размер записи | | | 1652 |

Таблица 14 – Таблица «Медицинский продукт состав»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **id** | **Уникальный идентификатор** | **int** | **4** |
| Name | Название компонента в составе | nvarchar(50) | 102 |
| *MedicinalProductId* | *Уникальный идентификатор медицинского препарата* | *int* | *4* |
| Размер записи | | | 110 |

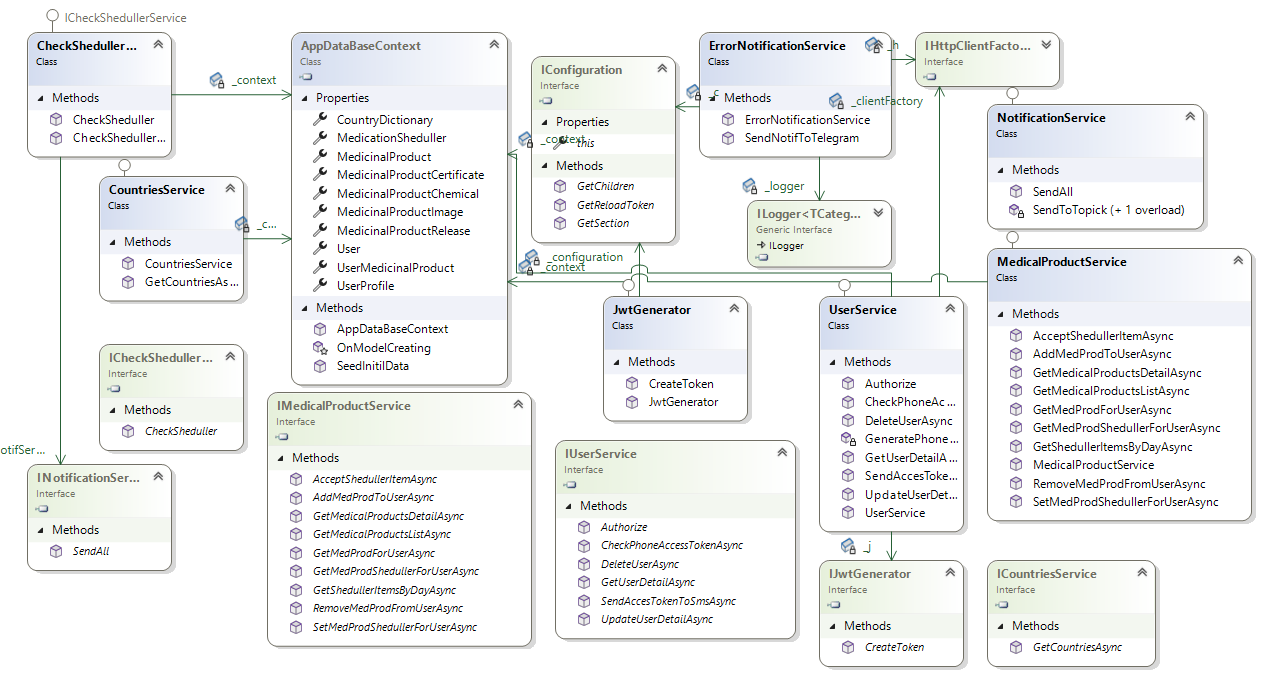


Рисунок 34 – Диаграмма классов инфраструктурного слоя системы

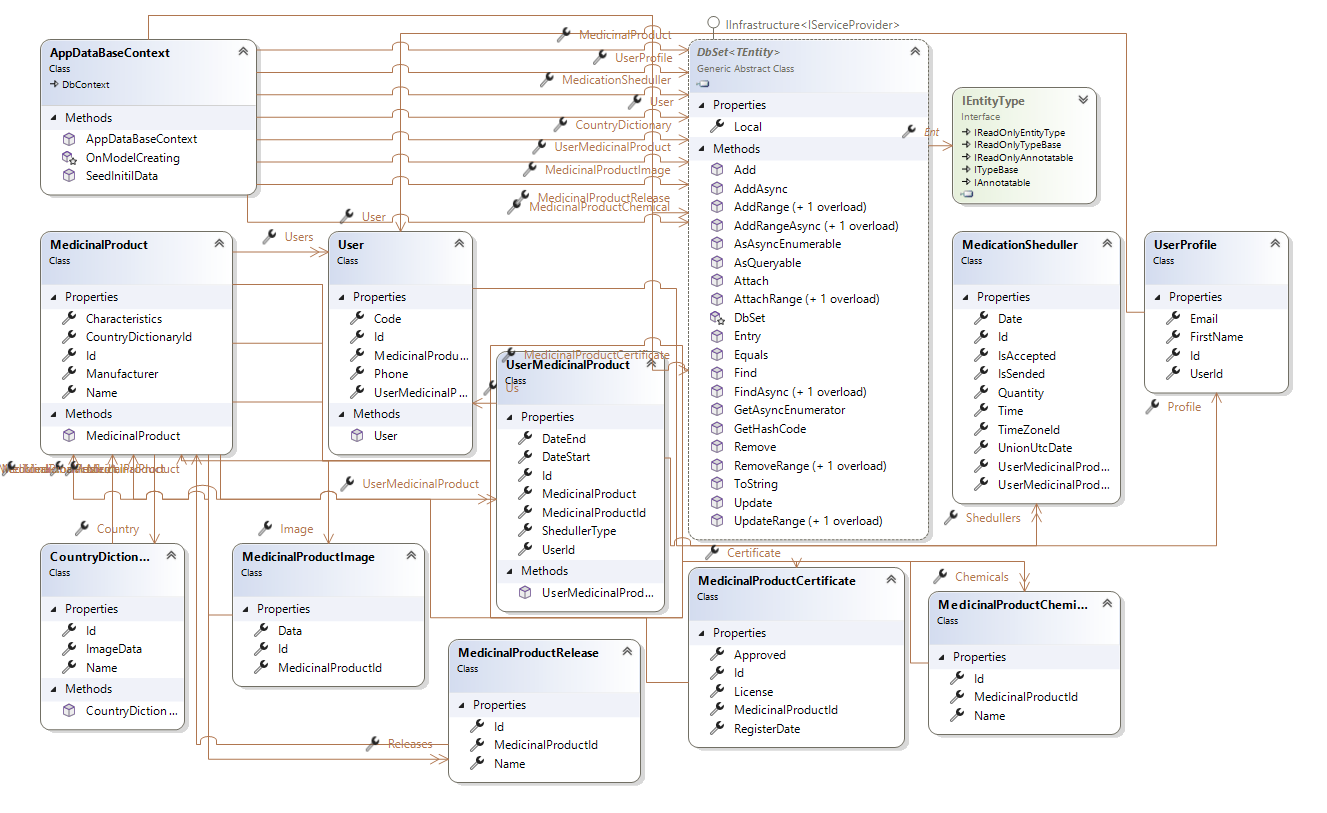


Рисунок 35 – Диаграмма классов доменного слоя системы

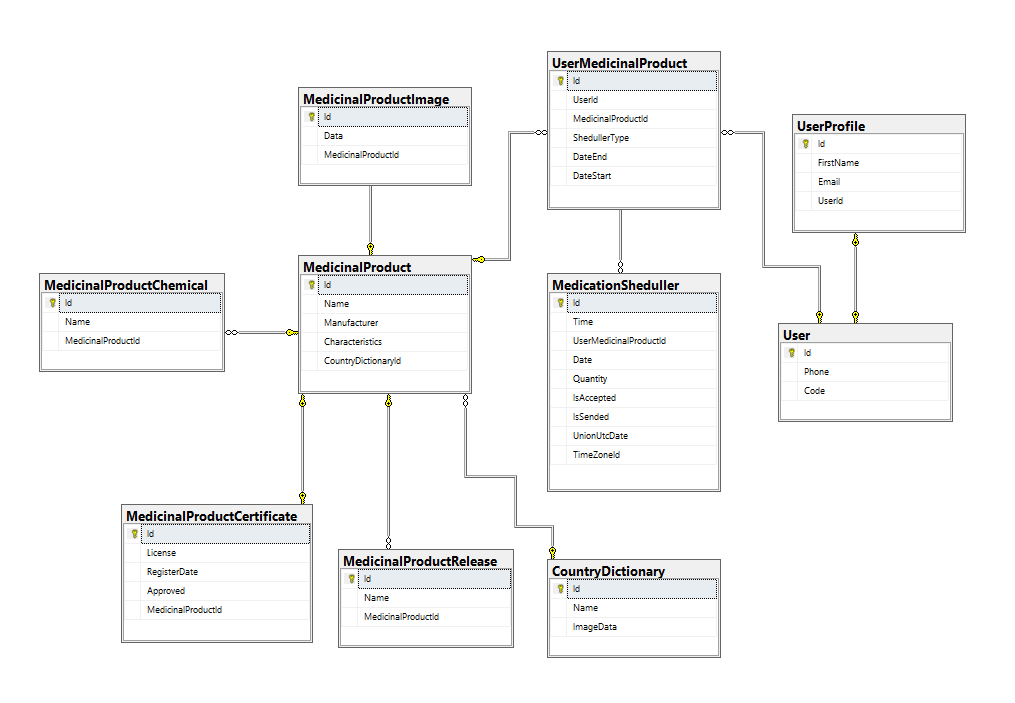


Рисунок 36 – Физическая модель базы данных

Таблица 15 – Таблица «Напоминание»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **id** | **Уникальный идентификатор** | **int** | **4** |
| Time | Время приема препарата | time(7) | 7 |
| *UserMedicinalProductId* | *Уникальный идентификатор медицинского продукта пользователя* | *int* | *4* |
| Date | Дата приема препарата | datetime2(7) | 7 |
| Quantity | Дозировка приема препарата | float | 4 |
| IsAccepted | Флаг принят ли препарат пользователем | bit | 1 |
| IsSended | Флаг было ли отправлено сообщение пользователю | bit | 1 |
| UnionUtcDate | Дата + время приема преобразованные в UTC | datetime2(7) | 7 |
| TimeZoneId | Временная зона пользователя | nvarchar(10) | 22 |
| Размер записи | | | 57 |

Таблица 16 – Таблица «Медицинский продукт»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **id** | **Уникальный идентификатор** | **int** | **4** |
| Name | Название медицинского препарата | nvarchar(20) | 42 |
| Manufacturer | Производитель медицинского препарата | nvarchar(50) | 102 |
| Characteristics | Прочие характеристики медицинского препарата | nvarchar(200) | 402 |
| *CountryDictionaryId* | *Уникальный идентифкатор словаря стран* | *int* | *4* |
| Размер записи | | | 554 |

Таблица 17 – Таблица «Медицинский продукт сертификат»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **id** | **Уникальный идентификатор** | **int** | **4** |
| License | Номер лицензии, выданый медицинскому препарату | nvarchar(20) | 42 |
| RegisterDate | Дата регистрации медицинского препарата | datetime2(7) | 7 |
| Approved | Статус регистрации медицинского препарата | nvarchar(10) | 22 |
| *MedicinalProductId* | *Уникальный идентификатор медицинского препарата* | *int* | *4* |
| Размер записи | | | 79 |

Таблица 18 – Таблица «Медицинский продукт изображение»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **id** | **Уникальный идентификатор** | **int** | **4** |
| Data | Изображение медицинского препарата в двоичном формате | varbinary(MAX) | 2147483648 (max)  1 550 (avg) |
| *MedicinalProductId* | *Уникальный идентификатор медицинского препарата* | *int* | *4* |
| Размер записи | | | 1558 |

Таблица 19 – Таблица «Медицинский продукт формы выпуска»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **id** | **Уникальный идентификатор** | **int** | **4** |
| Name | Название формы выпуска медицинского препарата | nvarchar(50) | 102 |
| *MedicinalProductId* | *Уникальный идентификатор медицинского препарата* | *int* | *4* |
| Размер записи | | | 110 |

Таблица 20 – Таблица «Медицинский продукт пользователь»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **id** | **Уникальный идентификатор** | **int** | **4** |
| UserId | *Уникальный идентификатор пользователя* | int | 4 |
| *MedicinalProductId* | *Уникальный идентификатор медицинского препарата* | *int* | *4* |
| ShedullerType | Тип переодичности напоминания о приеме лекарства | int | 4 |
| DateEnd | Дата окончания приема лекарства | datetime2(7) | 7 |
| DateStart | Дата начала приема лекарства | datetime2(7) | 7 |
| Размер записи | | | 30 |

Таблица 21 – Таблица «Пользователь»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **id** | **Уникальный идентификатор** | **int** | **4** |
| Phone | Номер телефона пользователя | nvarchar(12) | 26 |
| Code | Код подтверждения для авторизации | nvarchar(4) | 10 |
| Размер записи | | | 40 |

Таблица 22 – Таблица «Профиль пользователя»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **id** | **Уникальный идентификатор** | **int** | **4** |
| FirstName | Имя пользователя | nvarchar(20) | 42 |
| Email | Электронная почта пользователя | nvarchar(20) | 42 |
| *UserId* | *Уникальный идентификатор пользователя* | *int* | *4* |
| Размер записи | | | 92 |

* 1. Выбор и обоснование комплекса технических средств
     1. Расчет объема занимаемой памяти клиентской части системы

Для расчёта необходимого объёма свободной внешней памяти на устройстве воспользуемся формулой:

VЗП = VОС + VПР + VДП + VРК,

где VОС – объем памяти, занимаемый операционной системой (операционная система Android 12.0, VОС = 2 Гб);

VПР – объём памяти, занимаемый самим приложением (VПР = 20 Мб);

VДП – объём памяти, занимаемый пользовательскими данными   
(VДП = 103 Мб);

VРК – объём памяти, занимаемый кэшем (VРК = 290 Кб).

Таким образом, суммарный объем внешней памяти составит:

VЗП = 2Гб + 20 Мб + 103 Мб + 290 Кб ~ 2.1 Гб.

Расчет объема ОЗУ

Для расчета необходимого объема ОЗУ воспользуемся следующей формулой:

VОЗУ = VОС + VПР + VБД,

где VОС – ОЗУ, занимаемое операционной системой (2 Гб);

VПР – ОЗУ, которое займет само приложение (не превысит 50 Мб).

Суммарные объемы ОЗУ составит:

VОЗУ = 2 Гб + 50 Мб ~ 2 Гб.

Таким образом, 2.1 Гб оперативной памяти можно счесть минимально необходимым для функционирования системы.

* + 1. Расчет объема занимаемой памяти серверной части системы

Для расчета необходимого объема свободной внешней памяти, необходимой для функционирования системы, воспользуемся следующей формулой:

VЖД = VОС + VПР + VСПО + VБД,

где VОС – объем памяти, занимаемый операционной системой (операционная система Windows 10 Home 64 бит с пакетом обновлений SP1,   
VОС = 16 Гб);

VПР – объем памяти, занимаемый непосредственно файлами приложения (VПР = 29 Мб);

VСПО – объем памяти, занимаемый сопутствующим программным обеспечением (VСПО = 2 Гб);

VБД – объем памяти, занимаемый базой данных (всеми таблицами) при ее максимальном заполнении. Расчет этой составляющей приведен в таблице 23 (VБД = 0,4 Мб).

Таким образом, суммарный объем внешней памяти составит:

VЖД = 16 Гб + 29 Мб + 2 Гб + 0,4 Мб ~ 18 Гб.

Расчет объема ОЗУ

Для расчета необходимого объема ОЗУ воспользуемся следующей формулой:

VОЗУ = VОС + VПР + VБД,

где VОС – ОЗУ, занимаемое операционной системой (2 Гб);

VПР – ОЗУ, которое займет само приложение (не превысит 400 Мб);

VБД – объем данных из базы, который может быть одновременно загружен в оперативную память (дадим ему оценку сверху в 100 Мб).

Таблица 23 – Расчет объема внешней памяти, необходимой для хранения БД

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Размер записи (байт) | Максимум записей | Всего (байт) |
| Словарь стран | 1652 | 21 | 69384 |
| Медицинский продукт состав | 110 | 220 | 24200 |
| Напоминание | 57 | 600 | 34200 |
| Медицинский продукт | 554 | 95 | 52630 |
| Медицинский продукт сертификат | 79 | 95 | 7505 |
| Медицинский продукт изображение | 1558 | 95 | 148010 |
| Медицинский продукт формы выпуска | 110 | 609 | 66990 |
| Медицинский продукт пользователь | 30 | 90 | 2700 |
| Пользователь | 40 | 30 | 1200 |
| Профиль пользователя | 92 | 30 | 2760 |
| Итого | | | 409 579 |

Суммарные объемы ОЗУ составит:

VОЗУ = 2 Гб + 400 Мб + 100 МБ ~ 2.5 Гб.

Таким образом, 3 Гб оперативной памяти можно счесть минимально необходимым для функционирования системы.

* + 1. Минимальные требования, предъявляемые к системе

Для корректного функционирования клиентской части системы необходимо:

* тип ЭВМ: мобильное устройство;
* архитектура процессора: x86-64/arm совместимый;
* объем ОЗУ – не менее 2 Гб;
* объем свободного дискового пространства – не менее 6 Гб;
* экран с плотностью пикселей 440dpi и более;
* операционная система: Android 12 и выше – API 21 и выше / IOS 14.2 и выше.

Для корректного функционирования серверной части системы необходимо:

* тип ЭВМ: IBM PC совместимый;
* архитектура процессора: x86-64/arm совместимый;
* объем ОЗУ – не менее 4 Гб;
* объем свободного дискового пространства – не менее 50 Гб;
* процессор – не менее 4 ядер с тактовой частотой 2,5 ГГц;
* операционная система Windows/ любой Linux дистрибутив.

ЗАКЛЮЧЕНИЕ

В процессе выполнения лабораторного практикума было разработано мобильное приложение для поиска лекарств из открытых источников и напоминания об их приеме, позволяющее каждому желающему составлять свою собственную библиотеку лекарств и настраивать напоминания, чтобы не забыть принять необходимое лекарство.

В первом разделе приведены основные понятия предметной области, характеристики систем-аналогов и результаты их сравнительного анализа. Сформулирована постановка задачи.

Во втором разделе разработаны прототипы экранных форм. Были спроектированы диаграммы вариантов использования и классов. Была разработана логическая модель данных. Также были разработаны алгоритмы обработки данных и выбран комплекс программных средств для их реализации.

В третьей главе приведено описание разработанного пользовательского интерфейса системы и его возможностей, построены диаграммы реализации: развертывания, классов. Была разработана физическая модель данных, приведены выбор и обоснование комплекса технических средств, приведены ресурсные отчеты.

Разработанная система будет полезна людям, которым нужно напоминать о приеме лекарств с определённой периодичностью.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Буч Г., Рамбо Д., Якобсон А. Язык UML. Руководство пользователя. Изд. 2-е. М.: ДМК Пресс, 2006. 546 с.
2. Леоненков, А.В. Нотация и семантика языка UML [Электронный ресурс]/ А.В. Леоненков. – Интернет-университет информационных технологий. URL: http://www.intuit.ru/department/pl/umlbasics (дата обращения: 09.10.2022).
3. Лекарственное средство [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Лекарственное\_средство (дата обращения: 08.10.2022).
4. Аптечные сети [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/aptechnye-seti-1 (дата обращения: 08.10.2022).
5. Google Play [Электронный ресурс]. – URL: https://play.google.com/store/apps/details?id=com.careclinicsoftware.careclinic&gl=NL (дата обращения: 08.10.2022).
6. App Store [Электронный ресурс]. – URL: <https://apps.apple.com/ru/app/id1248342340> (дата обращения: 08.10.2022).
7. Wikipedia [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Клиент\_—\_сервер (дата обращения: 08.10.2022).
8. RESTful API [Электронный ресурс]. – URL: https://aws.amazon.com/ru/what-is/restful-api/ (дата обращения: 11.10.2022).
9. Swagger [Электронный ресурс]. – URL: https://swagger.io/docs/specification/about/ (дата обращения: 11.10.2022).
10. HTTP [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/HTTP (дата обращения: 11.10.2022).
11. Структурная схема [Электронный ресурс] – URL: https://dic.academic.ru/dic.nsf/ruwiki/1642106 (дата обращения: 08.10.2022).
12. Разработка пользовательских интерфейсов [Электронный ресурс] – URL: <http://pandia.ru/text/78/247/74988.php> (дата обращения 08.10.22).
13. UML-диаграммы. Виды UML диаграмм [Электронный ресурс]. URL: <https://www.syl.ru/article/206012/new_uml-diagramma-vidyi-diagramm-uml> (дата обращения: 09.10.2022).
14. Построение диаграммы классов [Электронный ресурс]. URL: https://flexberry.github.io/ru/gpg\_class-diagram.html (дата обращения: 09.10.2022).
15. Логическая модель базы данных [Электронный ресурс]. URL: https://studfile.net/preview/2083751/page:2/ (дата обращения: 09.10.2022).
16. Алгоритмы [Электронный ресурс]. URL: https://otus.ru/nest/post829 (дата обращения: 10.10.2022).
17. Язык программирования C#: краткая история, возможности и перспективы [Электронный ресурс] // Что такое C# и где он используется: обзор основных возможностей. URL: https://timeweb.com/ru/ community/articles/chto-takoe-csharp (дата обращения: 10.10.2022).
18. Microsoft Visual Studio [Электронный ресурс] // Википедия: электрон. энциклопедия. 2001-2021. URL: https://ru.wikipedia.org/wiki/ Microsoft\_Visual\_Studio (дата обращения: 10.10.2022).
19. Microsoft SQL Server [Электронный ресурс] URL: <https://www.allware.ru/microsoft-sql-server-127> (дата обращения: 10.10.2022).
20. .NET MAUI [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/maui/what-is-maui (дата обращения: 10.10.2022).
21. .NET MAUI [Электронный ресурс] URL: https://helpiks.org/8-15388.html (дата обращения: 13.10.2022).
22. Диаграммы реализации [Электронный ресурс] URL: [Диаграммы реализации (helpiks.org)](https://helpiks.org/8-15388.html) (дата обращения: 06.11.2022).
23. Диаграмма развертывания [Электронный ресурс] URL: [Диаграммы реализации (helpiks.org)](https://helpiks.org/8-15388.html) (дата обращения: 06.11.2022).

ПРИЛОЖЕНИЕ А  
Руководство пользователя

А.1 Назначение системы

Данная система предназначена для поиска лекарств из открытых источников и составления графика приёма.

А.2 Условия работы системы

Для корректной работы системы необходимо наличие соответствующих программных и аппаратных средств.

Требования к техническому обеспечению серверной части системы:

* тип ЭВМ – IBM PC совместимый;
* архитектура процессора: x86-64/arm совместимый;
* объем ОЗУ – не менее 4 Гб;
* объем свободного дискового пространства – не менее 50 Гб;
* процессор – не менее 4 ядер с тактовой частотой 2,5 ГГц;
* наличие подключения к сети Интернет.

Требования к программному обеспечению серверной части системы:

* тип операционной системы – Windows 7 и выше/Linux;
* NET Runtime 7.0;
* СУБД – SQL Server 15.0.

Требования к техническому обеспечению клиентской части системы:

* тип ЭВМ – мобильное устройство;
* архитектура процессора: x86-64/arm совместимый;
* объем ОЗУ – не менее 2гб;
* объем свободного дискового пространства – не менее 6 Гб;
* экран с плотностью пикселей 440dpi и более;
* экран с разрешающей способностью не ниже 1080 х 2340;
* манипулятор – сенсорный экран;
* наличие подключения к сети Интернет.

Требования к программному обеспечению клиентской части системы:

* тип операционной системы – Android 12 и выше – API 21 и выше / IOS 14.2 и выше.

А.3 Установка системы

Клиентское приложение поставляется в формате архивных файлов для конкретной платформы – apk для Android и ipa для IOS. Данные файлы необходимо установить на смартфон через системные службы. Так же установка приложения доступна через соответствующие магазины приложений – GooglePlay для Android и AppStore для IOS.

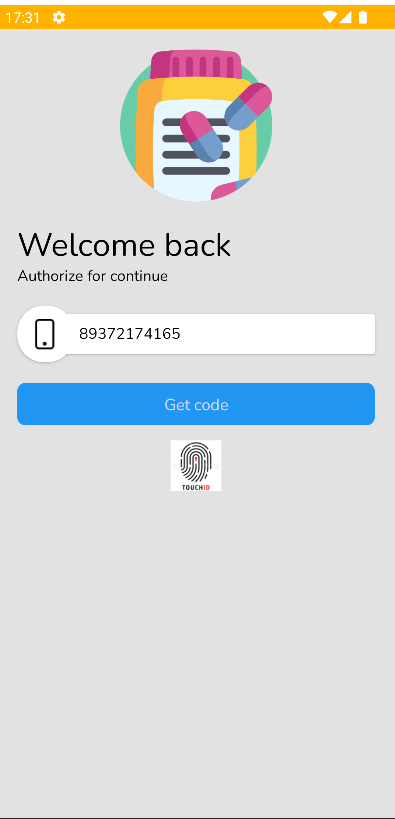
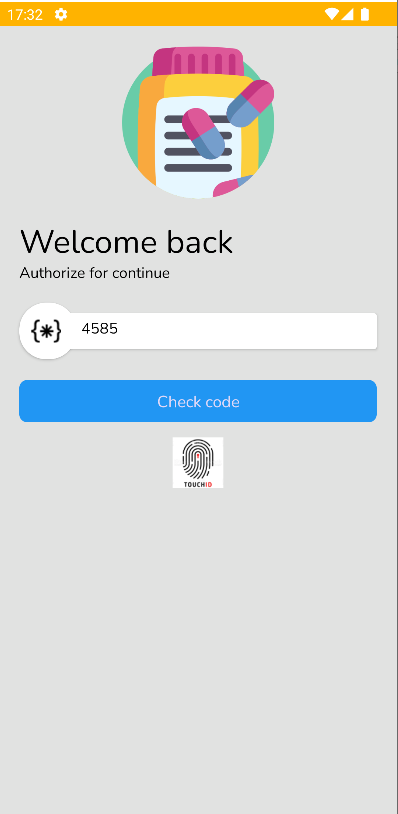
Серверная часть приложения поставляется в виде zip-архива. Данный файл необходимо распаковать в директорию на жестком диске и настроить веб сервер на работу с данной директорией. Запускаемым файлом является файл AppService.exe.

А.4 Работа с системой

При запуске мобильного приложения открывается экран авторизации (рисунок А.1). Регистрация происходит автоматически при первой авторизации в приложении пользователя.

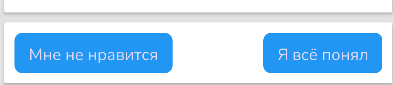
Чтобы авторизоваться в приложении, пользователь должен ввести номер телефона в специальное поле и нажать на кнопку «Get code». После чего ожидает СМС с кодом, вводит полученный код в специальное поле и нажимает на кнопку «Check code».

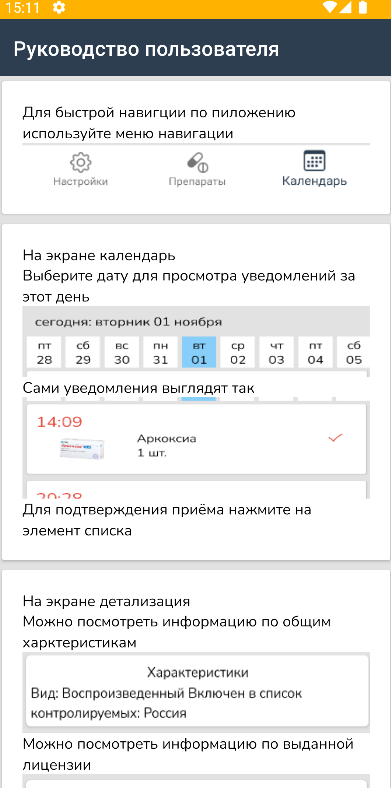
Во время последующих авторизаций пользователю будет доступна упрощённая авторизация через встроенные механизмы touchId и faceId на платформах Android и IOS соответственно.

   
Рисунок А.1 – Экран авторизации

Если пользователь впервые пользуется приложением, то после авторизаии откроется краткая инструкция по использование приложения на экране «Руководство пользователя» (рисунок А.2).

Пользователь может продолжить работу с приложением, нажав на кнопку «Я всё понял» или выйти из него нажав на кнопку «Мне не нравится» (рисунок А.3).

  
Рисунок А.3 – Действия на экране «Руководство пользователя»

  
Рисунок А.2 – Экран «Руководство пользователя»

После авторизации и/или ознакомления с руководством пользователя система открывает экран «Календарь» (рисунок А.4). В верхней части экрана показана дата на сегодняшний день. При нажатии на эту дату приложение автоматически выберет дату сегодняшнего дня из списка и отобразит все имеющиеся напоминания. Ниже отображается динамический список прошедших и будущих дат, при выборе даты из этого списка отображаются все напоминания за этот день.

На этом же экране пользователь может подтвердить прием препарата, нажав на конкретное напоминание.

Для быстрой навигации по приложению на каждом экране в нижней части присутствует специальное навигационное меню (рисунок А.5).

Каждая кнопка навигации перенаправляет пользователя на соответствующий экран в приложении.

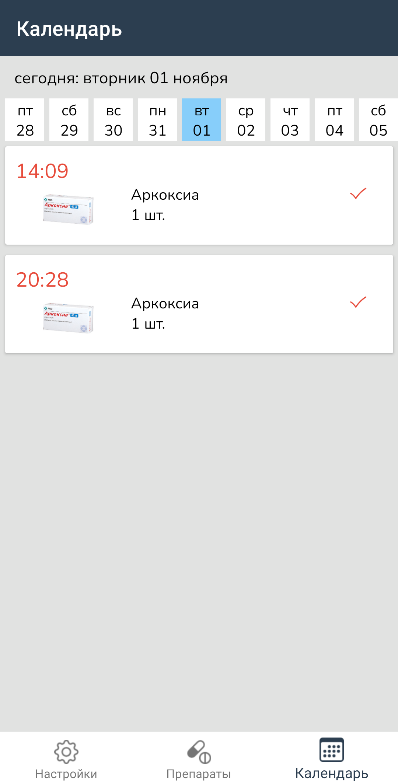
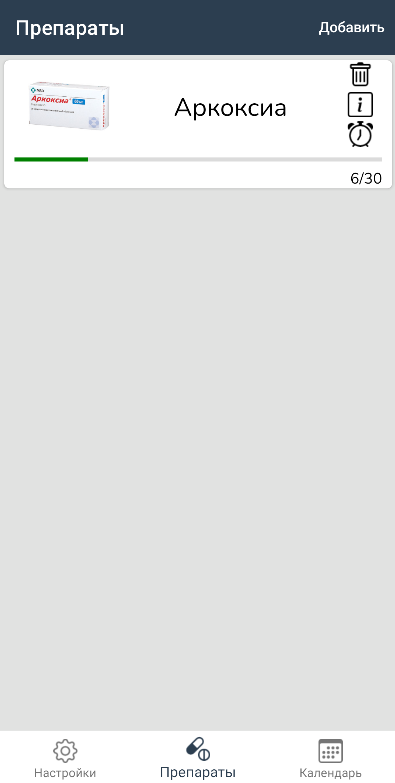
  
Рисунок А.4 – Экран «Календарь»

  
Рисунок А.5 – Навигационное меню

На рисунке А.6 изображен экран «Препараты». На данном экране показан список всех лекарственных средств, добавленных пользователем в свою билиотеку. Для того чтобы удалить запись, необходимо нажать на кнопку с изображением . Для того чтобы просмотреть детализацию по препарату, необходимо нажать на кнопку . Для того чтобы настроить напоминания о приеме препарата необходимо нажать на кнопку .

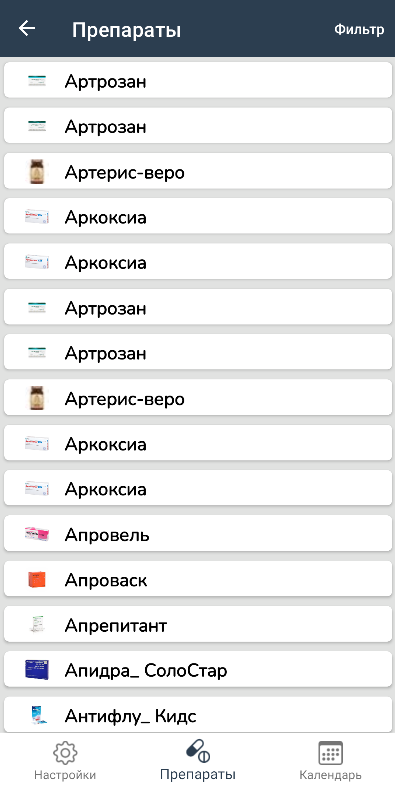
  
Рисунок А.6 – Экран «Препараты»

Под каждым лекарственным препаратом отображается полоса прогресса по принятию лекарств.

Для того чтобы добавить новый лекарственный препарат в свою библиотеку, необходимо нажать на кнопку «Добавить», после чего открывается экран поиска лекарственных средств (рисунок А.7).

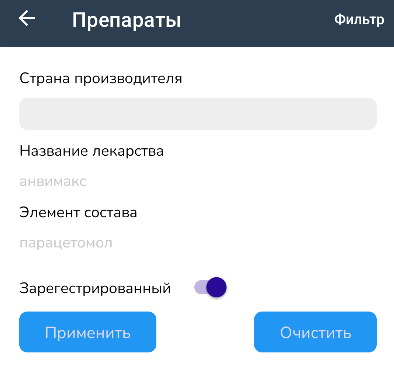
На данном экране отображается динамический список всех лекарственных средств, который подгружается автоматически из БД.

Для добавления препарата в свою библиотеку пользователю нужно нажать на необходимый препарат и, удерживая нажатым, потянуть элемент списка вправо.

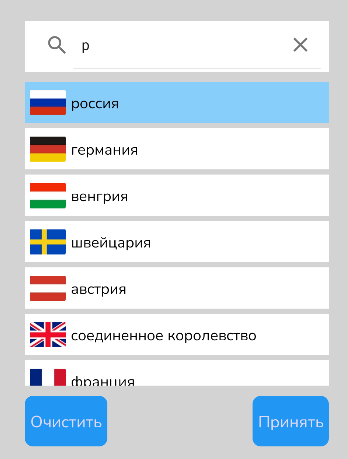
  
Рисунок А.7 – Экран поиска лекарственных средств

Для более удобного и быстрого поиска необходимого лекарственного препарата предусмотрена система фильтров для того, чтобы меню фильтров отобразилось нужно нажать на кнопку «Фильтр». После чего откроется специальная форма (рисунок А.8). На данной форме можно указать страну производителя, название лекарства, элемент состава, а также установить флаг наличия сертификата у препарата.

Чтобы применить фильтр, необходимо нажать на кнопку «Применить», а для того чтобы сбросить фильтры, на кнопку «Очистить».

  
Рисунок А.8 – Фильтр на экране поиска лекарственных средств

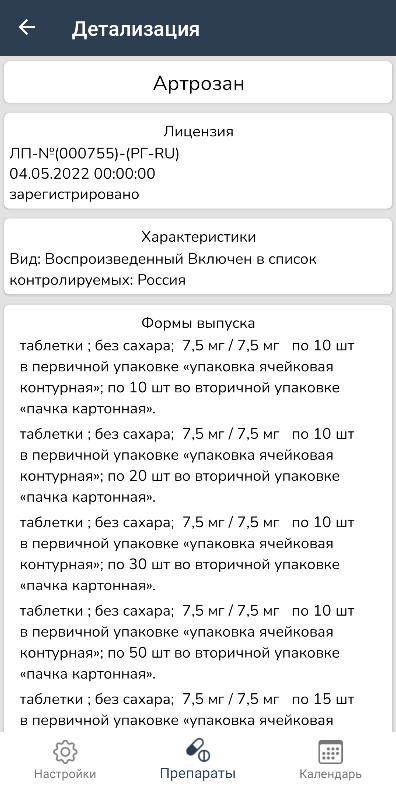
Выбор страны производителя осуществляется через специальное всплывающее меню (рисунок А.9).

  
Рисунок А.9 – Выбор страны производителя в фильтре

Для быстрого поиска страны в списке предусмотрено специальное поле, по которому будет отфильтрован список.

Для того чобы подтвердиь выбор страны, нужно нажать кнопку «Готово», а для очистки – кнопку «Очистить».

На рисунке А.10 изображен экран с детализацией лекарственного препарата. На данном экране показаны все имеющиеся в системе характеристики лекарственного препарата, а так же ссылки для перехода в интернет-магазины для покупки препарата.

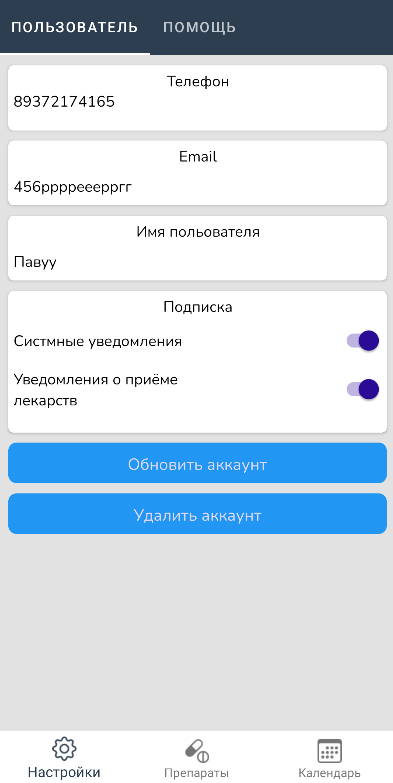
  
Рисунок А.10 – Экран с детализацией лекарственного средства

На рисунке А.11 изображен экран «Пользователь». На данном экране отображены личные данные пользователя с возможность их изменения.

Для изменения доступны поля «Email» и «Телефон». Чтобы применить изменения, нужно внести изменения в соответствующие поля и нажать на кнопку «Обновить аккаунт».

При нажатии на кнопку «Удалить аккаунт» произойдет полная очистка всех пользовательских данных.

Также на данном экране предоставлены функции отписки/подписки на оповещения посредством соответствующих переключателей.

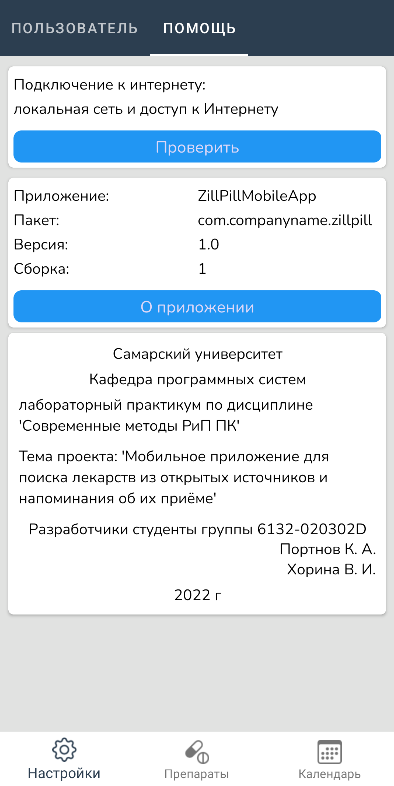
  
Рисунок А.11 – Экран «Пользователь»

На рисунке А.12 изображен экран «Помощь». На данном экране отображена вся основная информация о приложении и разработчиках, а также присутствует индикатор наличия интернет соединения.

Для того чтобы обновить статус индикатора наличия интернета, необходимо нажать на кнопку «Проверить».

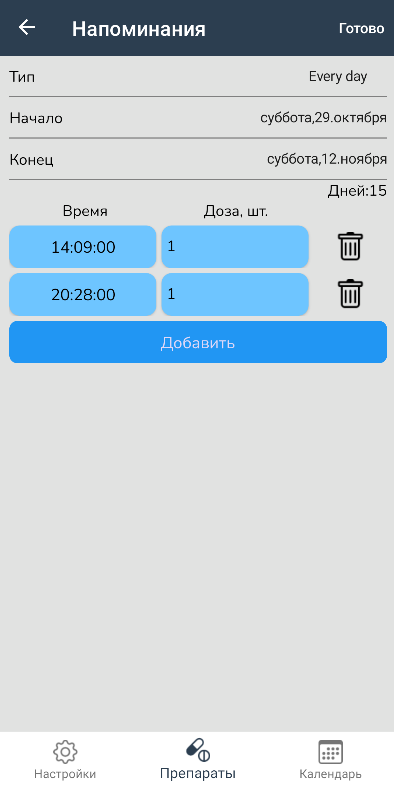
Для того чтобы посмотреть информацию о приложении, необходимо нажать на кнопку «О приложении».

На рисунке А.13 изображен экран «Напоминания». На данном экране доступны функции по созданию напоминаний для конкретного лекарственного препарата.

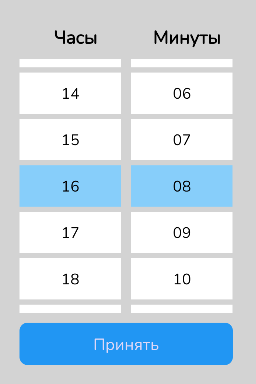
  
Рисунок А.12 – Экран «Помощь»

Для создания напоминаний имеется возможность выбрать тип периодичности (каждый день или в определенный день недели), дату начала и конца приема. В зависимости от выбранного типа периодичности изменяется отображаемая форма, и добавляются дополнительные элементы управления.

Для добавления нового напоминания нужно нажать на кнопку «Добавить» и указать время приема и дозировку препарата. Чтобы удалить напоминание, нужно нажать на кнопку .

  
Рисунок А.13 – Экран «Напоминания»

Время приёма препарата указывается через специальное всплывающее меню (рисунок А.14).

  
Рисунок А.14 – Выбор времени приёма препарата

В левой колонке на выбор предоставляется часы, а в правой минуты, после выбора требуемого времени нужно нажать кнопку «Принять».

Для завершения работы с напоминаниями и сохранения всех изменений, необходимо нажать на кнопку «Готово», после чего система открывает экран со список лекарственных препаратов пользователя.

На этом же экране у пользователя есть возможность в любой момент вносить изменения в ранее созданные напоминания, открывая данный экран для любого лекарственного средства из своего списка лекарств.

ПРИЛОЖЕНИЕ Б   
Листинг модулей программы

using Microsoft.EntityFrameworkCore;

using ZillPillService.Domain.DTO.MedicalProduct;

using ZillPillService.Domain.DTO.Shedullers;

using ZillPillService.Domain.Exceptions;

using ZillPillService.Domain.Query;

using ZillPillService.Domain.Query.User;

using ZillPillService.Infrastructure.Context;

using ZillPillService.Infrastructure.Entities;

using ZillPillService.Infrastructure.ServicesContract;

namespace ZillPillService.Application.Services

{

public class MedicalProductService : IMedicalProductService

{

private readonly AppDataBaseContext \_context;

public MedicalProductService(AppDataBaseContext context)

{

\_context = context;

}

public async Task<List<MedicalProductDto>> GetMedicalProductsListAsync(

GetFilteredMedicalProductQuery filter, int offset, int limit, CancellationToken ct)

{

var query = \_context.MedicinalProduct

.Include(x => x.Chemicals)

.Include(x => x.Certificate)

.Include(x => x.Image)

.OrderByDescending(x => x.Id)

.AsQueryable();

if (filter.CountryId is Int32 countryId)

query = query.Where(x => x.CountryDictionaryId == countryId);

if (filter.ProductName is String productName)

query = query.Where(x => x.Name.ToLower().Contains(productName.ToLower()));

if (filter.WithCertificate is bool withCertificate)

{

if (withCertificate)

query = query.Where(x => x.Certificate.Approved.Equals("зарегистрировано"));

else

query = query.Where(x => !x.Certificate.Approved.Equals("зарегистрировано"));

}

if (filter.ChemicalName is String chemicalName)

query = query.Where(x => x.Chemicals.Any(y => y.Name.ToLower().Contains(chemicalName.ToLower())));

var entities = await query

.Skip(offset)

.Take(limit)

.Select(x => new MedicalProductDto(x.Id, x.Name, x.Image.Data))

.ToListAsync(ct);

return entities;

}

public async Task<MedicalProductDetailDto> GetMedicalProductsDetailAsync(

int productId, CancellationToken ct)

{

var entity = await \_context.MedicinalProduct

.Where(x => x.Id == productId)

.Include(x => x.Image)

.Include(x => x.Certificate)

.Include(x => x.Releases)

.Include(x => x.Chemicals)

.Select(x => new MedicalProductDetailDto()

{

ProductId = x.Id,

ImageData = x.Image.Data,

Name = x.Name,

Characteristics = x.Characteristics,

Releases = x.Releases.Select(x => x.Name).ToList(),

Chemicals = x.Chemicals.Select(x => x.Name).ToList(),

Certificate = new MedicalProductCertificateDto(x.Certificate.License, x.Certificate.RegisterDate, x.Certificate.Approved)

})

.FirstOrDefaultAsync(ct);

if (entity == null) throw new ApiException("Product not found");

return entity;

}

#region user

public async Task<List<UserMedicalProductDto>> GetMedProdForUserAsync(

int userId, CancellationToken ct)

{

var entities = await \_context.UserMedicinalProduct

.Where(x => x.UserId == userId)

.Include(x => x.MedicinalProduct)

.ThenInclude(y => y.Image)

.Select(x => new UserMedicalProductDto(x.MedicinalProduct.Id, x.MedicinalProduct.Name, x.MedicinalProduct.Image.Data, x.Id))

.ToListAsync(ct);

foreach (var entity in entities)

{

var totalShedullers = await \_context.MedicationSheduller

.Where(x => x.UserMedicinalProductId == entity.RelationId)

.Select(x => x.IsAccepted)

.ToListAsync(ct);

entity.TotalToAccept = totalShedullers.Count();

entity.TotalAccepted = totalShedullers.Count(x => x);

entity.Progress = entity.TotalToAccept != 0 ? (double)entity.TotalAccepted / entity.TotalToAccept : 0.0;

}

return entities;

}

public async Task AddMedProdToUserAsync(int userId, int productId, CancellationToken ct)

{

var userProd = await \_context.UserMedicinalProduct

.Where(x => x.UserId == userId && x.MedicinalProductId == productId)

.FirstOrDefaultAsync(ct);

if (userProd != null)

throw new ApiException("dublicate product");

var product = await \_context.MedicinalProduct

.FirstOrDefaultAsync(x => x.Id == productId, ct);

if (product == null)

throw new ApiException("med product not found");

var user = await \_context.User

.FirstOrDefaultAsync(x => x.Id == userId, ct);

if (user == null)

throw new ApiException("user not found");

var newUserProd = new UserMedicinalProduct()

{

User = user,

MedicinalProduct = product

};

await \_context.UserMedicinalProduct.AddAsync(newUserProd, ct);

await \_context.SaveChangesAsync(ct);

}

public async Task RemoveMedProdFromUserAsync(int relationId, CancellationToken ct)

{

var entity = await \_context.UserMedicinalProduct

.Where(x => x.Id == relationId)

.FirstOrDefaultAsync(ct);

if (entity == null)

throw new ApiException("relation not found");

\_context.UserMedicinalProduct.Remove(entity);

await \_context.SaveChangesAsync(ct);

}

#endregion user

#region sheduller

public async Task<SetShedullerToUserDto> GetMedProdShedullerForUserAsync(

int relationId, CancellationToken ct)

{

var entity = await \_context.UserMedicinalProduct

.Include(x => x.Shedullers)

.Include(x => x.MedicinalProduct)

.Where(x => x.Id == relationId)

.FirstOrDefaultAsync(ct);

if (entity == null)

throw new ApiException("relation not found");

var dayOfWeeks = entity.Shedullers.Any() ? entity.Shedullers.Select(x => x.Date.DayOfWeek).Distinct().ToList() : new();

List<ShedullerItem> ShedullerItems = new();

if (entity.Shedullers.Any())

ShedullerItems = entity.Shedullers

.GroupBy(x => x.Date)

.First()

.Select(x => new ShedullerItem(x.Date, x.Time, x.Quantity, x.UnionUtcDate, x.TimeZoneId))

.ToList();

SetShedullerToUserDto result = new()

{

ShedullerType = entity.ShedullerType,

DateStart = entity.DateStart,

DateEnd = entity.DateEnd,

RelationId = entity.Id,

ProductId = entity.MedicinalProduct.Id,

DayOfWeeks = dayOfWeeks,

ShedullerItems = ShedullerItems

};

return result;

}

public async Task SetMedProdShedullerForUserAsync(

SetShedullerToUserDto query, CancellationToken ct)

{

var entity = await \_context.UserMedicinalProduct

.Include(x => x.Shedullers)

.Include(x => x.MedicinalProduct)

.Where(x => x.Id == query.RelationId)

.FirstOrDefaultAsync(ct);

var nowUTC = DateTime.Now.ToUniversalTime();

if (entity == null)

throw new ApiException("relation not found");

entity.ShedullerType = query.ShedullerType;

entity.DateEnd = query.DateEnd;

entity.DateStart = query.DateStart;

entity.Shedullers = query.ShedullerItems

.Select(x => new MedicationSheduller()

{

Date = x.Date,

Time = x.Time,

Quantity = x.Quantity,

IsAccepted = x.UnionUtcDate < nowUTC,

IsSended = x.UnionUtcDate < nowUTC,

UnionUtcDate = x.UnionUtcDate,

TimeZoneId = x.TimeZoneId,

})

.ToList();

\_context.UserMedicinalProduct.Update(entity);

await \_context.SaveChangesAsync(ct);

}

#endregion sheduller

#region sheduller items

public async Task<IEnumerable<ShedullerItemDetailDto>> GetShedullerItemsByDayAsync(int userId, DateTime date, CancellationToken ct)

{

var shedullers = await \_context.UserMedicinalProduct

.Where(x => x.UserId == userId)

.Include(x => x.MedicinalProduct)

.ThenInclude(x => x.Image)

.SelectMany(x => x.Shedullers.Where(x => x.Date == date))

.Select(x => new ShedullerItemDetailDto(x.Id, x.Date, x.Time, x.Quantity, x.IsSended, x.IsAccepted,

x.UserMedicinalProduct.MedicinalProduct.Name, x.UserMedicinalProduct.MedicinalProduct.Image.Data))

.ToListAsync(ct);

return shedullers;

}

public async Task AcceptShedullerItemAsync(int shedullerItemId, CancellationToken ct)

{

var entity = await \_context.MedicationSheduller

.FirstOrDefaultAsync(x => x.Id == shedullerItemId, ct);

if (entity == null)

throw new ApiException("sheduller not found");

entity.IsAccepted = true;

\_context.MedicationSheduller.Update(entity);

await \_context.SaveChangesAsync(ct);

}

#endregion sheduller items

}

}

using Microsoft.EntityFrameworkCore;

using ZillPillService.Domain.DTO.User;

using ZillPillService.Domain.Exceptions;

using ZillPillService.Domain.Query.User;

using ZillPillService.Infrastructure.Context;

using ZillPillService.Infrastructure.Entities;

using ZillPillService.Infrastructure.ServicesContract;

namespace ZillPillService.Application.Services

{

public class UserService : IUserService

{

private readonly AppDataBaseContext \_context;

private readonly IJwtGenerator \_jwtTokenService;

private readonly IHttpClientFactory \_clientFactory;

public UserService(

AppDataBaseContext context, IJwtGenerator jwtGenerator, IHttpClientFactory clientFactory)

{

\_context = context;

\_jwtTokenService = jwtGenerator;

\_clientFactory = clientFactory;

}

/// <summary>

/// send sms with code to user

/// </summary>

/// <param name="phone"></param>

/// <param name="ct"></param>

/// <returns></returns>

public async Task SendAccesTokenToSmsAsync(string phone, CancellationToken ct)

{

var user = await \_context.User

.FirstOrDefaultAsync(x => x.Phone.Equals(phone), ct);

if (user == null)

{

User newUser = new()

{

Phone = phone,

Profile = new UserProfile()

{

FirstName = null,

Email = null

}

};

await \_context.User.AddAsync(newUser, ct);

await \_context.SaveChangesAsync(ct);

user = await \_context.User

.FirstOrDefaultAsync(x => x.Phone.Equals(phone), ct);

}

var code = GeneratePhoneNumberTokenAsync();

user.Code = code;

\_context.User.Update(user);

await \_context.SaveChangesAsync(ct);

//var client = \_clientFactory.CreateClient("smsAreaApi");

//var message = $"код для доступа: {code}";

//Dictionary<string, string> queryParam = new()

//{

// {"number", $"{phone}"},

// {"text", $"{message}"},

// {"sign", "SMS Aero"}

//};

//var uri = QueryHelpers.AddQueryString(client.BaseAddress.AbsoluteUri, queryParam);

//var request = new HttpRequestMessage(HttpMethod.Get, uri);

//var response = await client.SendAsync(request, ct);

//var responseMessage = await response.Content.ReadAsStringAsync();

//if (!response.IsSuccessStatusCode)

// throw new ApiException("ошибка при отправке sms");

}

/// <summary>

/// check code from sms and user

/// </summary>

/// <param name="phone"></param>

/// <param name="code"></param>

/// <param name="ct"></param>

/// <returns></returns>

public async Task<LoginResponseDto> CheckPhoneAccessTokenAsync(

string phone, string code, CancellationToken ct)

{

var user = await \_context.User

.FirstOrDefaultAsync(x => x.Phone.Equals(phone), ct);

if (user == null)

throw new ApiException("пользователей не найден");

//if (code != user.Code)

// throw new ApiException("код не совпадает");

return await Authorize(user);

}

/// <summary>

/// authentication user on login/pass

/// </summary>

/// <param name="user"></param>

/// <returns></returns>

public async Task<LoginResponseDto> Authorize(

User user)

{

string token = \_jwtTokenService.CreateToken(user);

var result = new LoginResponseDto(token);

return result;

}

/// <summary>

/// generate 4-digit code

/// </summary>

/// <returns></returns>

private string GeneratePhoneNumberTokenAsync()

{

var rnd = new Random();

var start = rnd.Next(9, 99);

var end = DateTime.Now.Second;

return start.ToString() + end.ToString();

}

/// <summary>

/// get user detail

/// </summary>

/// <param name="userId"></param>

/// <param name="ct"></param>

/// <returns></returns>

/// <exception cref="ApiException"></exception>

public async Task<UserDetailDto> GetUserDetailAsync(int userId, CancellationToken ct)

{

var user = await \_context.User

.Include(x => x.Profile)

.FirstOrDefaultAsync(x => x.Id == userId);

if (user == null)

throw new ApiException("user not found");

UserDetailDto result = new()

{

Phone = user.Phone,

Email = user.Profile.Email,

FirstName = user.Profile.FirstName

};

return result;

}

/// <summary>

/// update user detail

/// </summary>

/// <param name="userId"></param>

/// <param name="query"></param>

/// <param name="ct"></param>

/// <returns></returns>

/// <exception cref="ApiException"></exception>

public async Task UpdateUserDetailAsync(int userId, UpdateUserDetailQuery query, CancellationToken ct)

{

var user = await \_context.User

.Include(x => x.Profile)

.FirstOrDefaultAsync(x => x.Id == userId);

if (user == null)

throw new ApiException("user not found");

user.Profile.Email = query.Email;

user.Profile.FirstName = query.FirstName;

\_context.User.Update(user);

await \_context.SaveChangesAsync(ct);

}

/// <summary>

/// delete user

/// </summary>

/// <param name="userId"></param>

/// <param name="ct"></param>

/// <returns></returns>

/// <exception cref="ApiException"></exception>

public async Task DeleteUserAsync(int userId, CancellationToken ct)

{

var user = await \_context.User

.FirstOrDefaultAsync(x => x.Id == userId);

if (user == null)

throw new ApiException("user not found");

\_context.User.Remove(user);

await \_context.SaveChangesAsync(ct);

}

}

}

using Microsoft.EntityFrameworkCore;

using ZillPillService.Domain.Query.Notification;

using ZillPillService.Infrastructure.Context;

using ZillPillService.Infrastructure.ServicesContract;

namespace ZillPillService.Application.Services

{

public class CheckShedullerService : ICheckShedullerService

{

private readonly AppDataBaseContext \_context;

private readonly INotificationService \_notifService;

public CheckShedullerService(

AppDataBaseContext context, INotificationService notifService)

{

\_context = context;

\_notifService = notifService;

}

public async Task CheckShedullersAsync(CancellationToken ct)

{

var nowUTC = DateTime.Now.ToUniversalTime();

var entities = await \_context.MedicationSheduller

.Include(x => x.UserMedicinalProduct)

.ThenInclude(x => x.User)

.Include(x => x.UserMedicinalProduct)

.ThenInclude(x => x.MedicinalProduct)

.Where(x => !x.IsSended && x.UnionUtcDate <= nowUTC)

.ToListAsync(ct);

if (entities.Any())

{

foreach (var entity in entities)

{

var topick = $"sheduller\_{entity.UserMedicinalProduct.User.Phone}";

NotificationQuery mesage = new()

{

Type = Domain.Model.NotificationTypeEnum.Sheduller,

Describle = $"Пора принимать {entity.UserMedicinalProduct.MedicinalProduct.Name}, {entity.Quantity} шт.",

Title = "Напоминание"

};

await \_notifService.SendNotifAsync(topick, mesage, ct);

entity.IsSended = true;

}

\_context.MedicationSheduller.UpdateRange(entities);

await \_context.SaveChangesAsync();

}

}

public async Task SendSystemMessgeAsync(string body, CancellationToken ct)

{

var titleToSend = "Системное сообщение";

var topick = $"system";

NotificationQuery mesage = new()

{

Type = Domain.Model.NotificationTypeEnum.System,

Describle = body,

Title = titleToSend

};

await \_notifService.SendNotifAsync(topick, mesage, ct);

}

}

}