МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему

Библиотека регистрации и идентификации пользователя для мобильного клиента

Студент _____ Забродина Ксения Андреевна
(подпись, дата) (ФИО полностью)

Руководитель _____ А.Н. Токарев
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ А.В. Кучин
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 11.06.2020 № 1)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«30» апреля 2020г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

1. Студент Забродина Ксения Андреевна гр. 16ВВ1 факультета вычислительной техники

2.Тема работы Библиотека регистрации и идентификации пользователя для мобильного клиента

Тема утверждена приказом ПГУ № 0335а/о от "30" апреля 2020 г.

3. Руководитель выпускной квалификационной работы бакалавра Токарев А.Н., к.т.н., доцент кафедры ВТ

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать и отладить кроссплатформенную библиотеку, предназначенную для сбора, обработки и хранения персональных данных пользователя. Библиотека должна поддерживать возможность асинхронной отправки запросов сторонним сервисам. Разработать и отладить мобильное приложение демонстрирующее, работоспособность разработанного программного модуля.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: современные технологии кроссплатформенной разработки, рассмотреть основные языки программирования для создания кроссплатформенных мобильных приложений и среды программирования. Необходимо изучить основные плагины и функционал фреймворка Flutter.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	10	30.04.2020 - 03.05.2020	
2.	Анализ требований	10	04.05.2020 - 06.05.2020	
3.	Выбор технологий, применяемых при проектировании и разработке	10	07.05.2020 - 10.05.2020	
4.	Разработка структуры проекта	10	11.05.2020 - 13.05.2020	
5.	Разработка БД	15	14.05.2020 - 17.05.2020	
6.	Разработка REST API	15	18.05.2020 - 21.05.2020	
7.	Разработка библиотеки регистрации и идентификации пользователя и приложения демонстрирующего её работу	20	22.05.2020 - 26.05.2020	
8.	Оформление пояснительной записки	10	27.05.2020 - 29.05.2020	

Дата выдачи задания "30" апреля 2020г.

Руководитель бакалаврской работы _____ Токарев А.Н.
(подпись, дата)

Задание к исполнению принял студент _____ Забродина К.А.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 150 лист, 21 рисунков, 2 таблицы, 12 источников, 2 приложения.

КРОСПЛАТФОРМЕННАЯ РАЗРАБОТКА, БД, REST, API, DART, FLUTTER.

Объектом исследования является библиотека регистрации и идентификации пользователя для мобильного клиента.

Предметом исследования является организация структуры кроссплатформенного мобильного ПО, а также механизмы сбора, обработки и хранения данных.

Цель работы – разработать библиотеку регистрации и идентификации пользователя для мобильного клиента.

Объектом разработки является библиотека, обеспечивающая выполнение следующих функций: сбор, хранение и обработку персональных данных пользователя. Хранение информации о пользователях, организациях и группах должно осуществляться в БД. Необходимо реализовать механизмы асинхронной обработки запросов от сторонних сервисов. Также требуется реализовать приложение, демонстрирующее функционал библиотеки.

В результате проведенной работы была разработана библиотека, удовлетворяющая поставленную задачу, а именно: разработана структура БД, разработан необходимый функционал для работы с персональными данными. Библиотека была отлажена и успешно интегрирована в приложение.

Перечень принятых сокращенийы

API (Application Programming Interface) – программный интерфейс приложения;

HTML (HyperText Markup Language) – язык гипертекстовой разметки;

CSS (Cascading Style Sheets) – язык таблиц стилей;

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста;

JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript;

REST (Representational State Transfer) – передача состояния представления;

SQL (Structured Query Language) – язык структурированных запросов;

UML (Unified Modeling Language) – унифицированный язык моделирования;

URL (Uniform Resource Locator– единообразный локатор ресурса;

XML (eXtensible Markup Language) – расширяемый язык разметки;

БД – База данных;

ПЭВМ (персональная электронно-вычислительная машина) – настольная микро-ЭВМ;

ВДТ (вольтодобавочный трансформатор) – устройство дифференциального тока;

ПО (Программное обеспечение) – множество программ;

ОС (Операционная система) – комплекс взаимосвязанных программ;

ES (Embedded Systems) – встраиваемая система;

SSL (Secure Sockets Layer) – уровень защищенных сокетов.

Содержание

Перечень принятых сокращений 5
Введение 7
1. Анализ предметной области 9
1.1. Анализ современных мобильных операционных систем 9
1.1.1. Android 10
1.1.2. iOS 14
1.2. Анализ современных технологий кроссплатформенной разработки 17
1.3. Выбор языка программирования 18
1.4. Выбор среды разработки 21
1.5. Выбор дополнительных инструментов разработки 23
1.6. Вывод 24
2. Реализация 26
2.1. Описание структуры проекта 26
2.2. Структура БД 28
2.3. Описание REST API 30
2.4. Описание классов 32
2.5. Реализация программы 34
2.6. Вывод 36
3. Инструкция пользователя 37
3.1. Описание пользовательского интерфейса 37
3.2. Вывод 44
4. Охрана труда 45
5. Заключение 48
Список используемых источников 49
Приложение А. Листинги программы на Dart 50
Приложение В. Описание REST API для работы с пользователями 141
Приложение С. Презентация 144

Введение

Мобильное устройство является неотъемлемой частью жизни любого современного человека, так как позволяет, эффективно решать большой спектр задач, оставаясь при этом мобильным. Многозадачность данного инструмента поддерживается благодаря наличию в нём разнообразного мобильного программного обеспечения, позволяющего упростить процесс коммуникации и обработки необходимой информации. Мобильное ПО в зависимости от своего назначения специализируется на обработку определённых вида информации.

Наиболее распространёнными являются мобильные приложения, работающие с персональными данными пользователя. Подобные приложения поддерживают различные механизмы по сбору, обработке и хранению необходимого набора данных. Реализация подобных механизмов является весьма трудозатратной, так как зачастую разработчику требуется учитывать необходимость работы с различными типами информации.

В данной работе была разработана универсальная библиотека для сбора, обработки, и хранения персональных данных. В состав библиотеки входят алгоритмы по обработке различных типов информации.

Объект исследования: библиотека регистрации и идентификации пользователя.

Предмет исследования: организация структуры кроссплатформенного мобильного ПО, а также механизмы сбора, обработки и хранения данных.

Степень изученности проблемы: теоретической основой для данного исследования послужили:

- современные кроссплатформенные технологии;
- протоколы и способы передачи данных в сети Интернет;
- методы организации и хранения данных.

Несмотря на то, что существует большое количество разнообразных приложений, осуществляющих сбор и обработку персональных данных, ещё не реализован универсальный кроссплатформенный модуль, позволяющий разработчику упростить процесс создания мобильного ПО.

Цель исследования: изучение существующих технологий кроссплатформенного программирования для мобильных устройств, а также компонентов для создания мобильного ПО - языков программирования, фреймворков, аппаратно-программных платформ, сред программирования, протоколов.

Задачи исследования:

- анализ современных мобильных операционных систем;
- анализ технологий кроссплатформенной разработки;
- разработка структуры библиотеки и приложения, демонстрирующего её работоспособность;
- разработка структуры БД для хранения персональных данных пользователя;
- тестирование работы созданной библиотеки регистрации и идентификации пользователя.

Практическая значимость заключается в разработке эффективного механизма работы с персональными данными пользователя. Внедрение данного модуля позволило упростить процесс сбора, хранения и обработки персональных данных пользователя.

1. Анализ предметной области

На начальном этапе разработки программных продуктов необходимо проанализировать интересующую предметную область. В данной главе рассмотрены современные стандарты и технологии, позволяющие реализовать универсальную библиотеку для работы с персональными данными пользователей. В данном случае осуществляется анализ современных мобильных операционных систем и технологий кроссплатформенной разработки для мобильных устройств, а также выбор архитектуры приложения и языка программирования.

1.1. Анализ современных мобильных операционных систем

Мобильная операционная система – это платформа, в состав которой входят взаимосвязанные системные программы, осуществляющие контроль, использование и распределения ресурсов мобильного устройства [1]. Отличительными особенностями мобильных ОС являются следующие факторы:

- адаптация под выполнение задач в условиях ограничений по памяти мобильных устройств;
- оптимизация работы системы с учетом низкой скорости процессора;
- адаптация под особенности экранов портативных устройств;
- обеспечение доступа к сетевым технологиям.

На данный момент существует множество различных операционных систем, согласно статистике, наиболее используемыми среди них являются:

- Android – 70,68%;
- iOS – 28,79%;
- Другие – 0,45%.

1.1.1. Android

Android является одной из самых популярных мобильных ОС, поскольку применяется в большинстве современных портативных устройств. Причиной востребованности данной операционной системы у разработчиков гаджетов является то, что она представлена в виде открытого исходного кода, что позволяет адаптировать её под особенности конкретного устройства [2].

Архитектура Android состоит из следующих уровней:

- приложения;
- каркас приложений;
- библиотеки;
- среда выполнения;
- ядро.

Уровень Приложения состоит из стандартного набора ПО и дополнительного стороннего, при этом для ОС не имеет значения основное это приложение или дополнительное. Приложения из стандартного набора, по желанию пользователя, могут быть заменены на альтернативные.

На уровне каркасных приложений находятся регулирующие жизненный цикл приложений основные службы системы Android, пакеты, ресурсы и т.д.

Данный уровень предоставляет разработчику полный доступ к API, используемому стандартными приложениями. При этом система построена таким образом, что позволяет заменять стандартные компоненты на пользовательские.

В состав данного уровня входят следующие элементы:

1. Менеджер деятельностей – это служба, предназначенная для регулирования жизненными циклами приложений и позволяющая получать доступ к системе навигации по истории работы с деятельностями.

2. Менеджер окон – это служба, предназначенная для сохранения окон приложения.
3. Контакт-провайдер – это служба, позволяющая обмениваться данными между приложениями.
4. Система представлений предназначена для создания графического интерфейса приложений и включающая, такие компоненты как поля ввода/вывода, кнопки, списки и т.д.
5. Менеджер пакетов – это служба, управляющая установкой пакетов прикладных программ.
6. Менеджер телефонии используется для предоставления различных утилит, управляющих телефонными номерами, и информации о сетевом типе, статусе подключения.
7. Менеджер ресурсов – это служба, применяемая для получения доступа к различным типам ресурсов.
8. Менеджер расположения – это служба, позволяющая получать информацию о местоположении устройства или запускать соответствующее ПО.
9. Менеджер уведомлений предназначен для отображения пользовательских уведомлений в строке состояния.

Следующим уровнем являются библиотеки. Он находится над ядром и содержит в себе набор библиотек, написанных на языках Си и Си++ и применяющихся различными компонентами системы.

По функциональному назначению библиотеки данного уровня можно разделить на следующие группы:

1. менеджер поверхностей;
2. системная библиотека;
3. функциональные библиотеки.

Менеджер поверхностей устроен так, что команды рисунка входят за кадром в битовые массивы, которые в последствии объединяются с другими битовыми массивами, в результате чего формируется изображение, отображающееся на экране устройства. Данный подход позволяет отображать на экране различные визуальные эффекты.

Системная библиотека была разработана компанией Google и получила название “bionic”. В состав библиотеки входит встроенная поддержка системных служб и регистрация системных событий.

Функциональными являются следующие библиотеки:

1. База данных SQLite – это библиотека, содержащая процессор баз данных, доступный для приложений. Хранение данных осуществляется в едином стандартном файле на выполняющем устройстве.
2. OpenGL ES (3D-графика) – это библиотека, предназначенная для работы с 3D-графикой. Данная библиотека использует аппаратный 3d-акселератор или трёхмерное ПО с высоким уровнем оптимизации для работы с растровой графикой.
3. Шрифты – это библиотека, работающая с растровыми и векторными шрифтами.
4. WebKit предназначена для функционирования встроенного web-браузера.
5. SGL (2D-графика) – это библиотека, осуществляющая работу с 2D-графикой.
6. SSL применяется для работы с сокетами. Основу библиотеки составляет протокол ssl.
7. Мультимедиа – это библиотека, основанная на packetvideo и предназначенная для воспроизведения аудио- и видео-форматов.

Среда выполнения состоит из двух элементов машины выполнения Dalvik и библиотеки ядра.

Dalvik – это виртуальная машина, относящаяся к классу регистровых машин, предназначенная для работы с процессорами мобильных устройств. Dalvik предоставляет разработчикам мобильного ПО удобный механизм разработки приложений и позволяет не заботится об объеме используемой памяти и о мощности процессора исполняемого устройства.

Библиотека ядра состоит из набора библиотек, обеспечивающих слой API и являющихся основной платформой разработки ПО для Android.

Уровень ядро является абстрактным слоем, отделяющим оборудование от остального программного стека. Основой ядра Android является ядро Linux, но несмотря на это, Android имеет существенные отличия, одним из которых является оснащённость дополнительными расширениями ядра.

Основными компонентами ядра являются:

1. драйвер межпроцессорного взаимодействия;
2. менеджер управления энергопотреблением;
3. набор драйверов оборудования, входящего в состав устройства.

Драйвер межпроцессорного взаимодействия – это драйвер, регулирующий процесс одновременной работы приложений и служб, при котором каждая из них функционирует в отдельном защищённом процессоре. При этом поддерживается взаимосвязь между процессорами и доступ к общим данным.

Менеджер управления энергопотреблением является драйвером, регулирующим потребления энергии батареи, с целью её экономии. Также данный драйвер осуществляет переход системы в “спящий режим”, если приложения и службы не находятся в активном состоянии.

Наглядное описание архитектуры ОС Android представлено на рисунке 1.

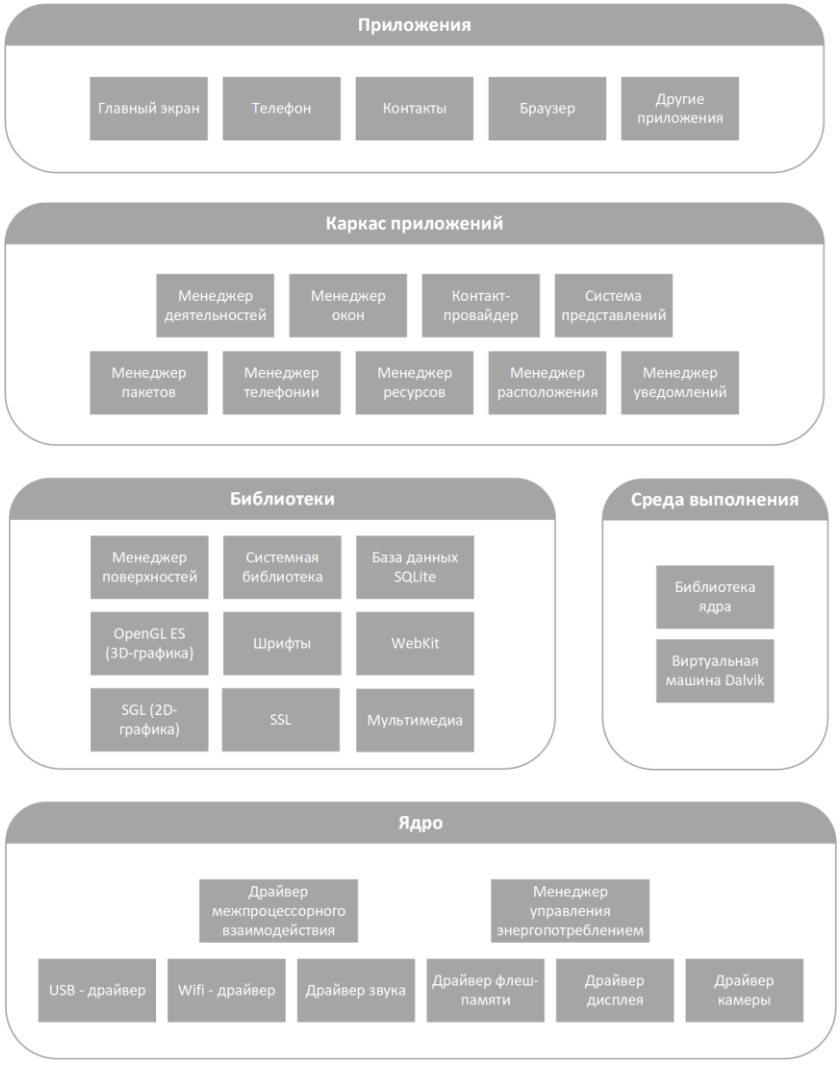


Рисунок 1 — схема архитектура ОС Android.

1.1.2. iOS

Операционная система iOS занимает второе по популярности место среди мобильных ОС на мировом рынке. Достоинством данной операционной системы является удобный интерфейс, а также высокий уровень безопасности. Однако, данная система не лишена недостатков, главный из которых является, необходимость приобретения дорогостоящего аккаунта разработчика, для размещения созданных приложений в AppStore.

Архитектура iOS состоит из следующих уровней:

- интерфейс;
- медиа;

- серверное ядро;
- ядро.

Уровень интерфейс состоит из большого количества элементов, основной целью которых является предоставление другим уровням доступа к данным, вводимым пользователем.

Следующий уровень называется медиа и содержит в себе инструменты, позволяющие обрабатывать медиа данные. В состав данного уровня входят следующие элементы:

1. Аудио – это сервис, предназначенный для записи, прослушивания, а также редактирования аудио записи.
2. OpenAL – это библиотека, предназначенная для работы с аудиоданными. Главная особенность данной библиотеки заключается в работе со звуком в 3D-пространстве и использовании эффектов EAX.
3. Видео является службой, позволяющей воспроизводить, перематывать, останавливать и редактировать видео записи.
4. OpenGL ES – это библиотека, предназначенная для работы с 3D-графикой. Аналог подобной библиотеки встречается в архитектуре ОС Android.
5. Quartz (2D) – это библиотека, предназначенная для работы с 2D-графикой и прорисовки двухмерных изображений.
6. Анимация – это служба, используемая для создания анимации. При создании анимации допускается как последовательный показ картинок, так и изменение свойств элементов интерфейса.

Уровень серверное ядро позволяет получить доступ к следующим сервисам:

1. Файловый диспетчер – это сервис, обладающий высоким уровнем интерфейса для получения доступа к файлам различных приложений.
2. Локация является службой, предназначенной для отслеживания местонахождения устройства.
3. Подобно Android, iOS имеет свою библиотеку, осуществляющую создание моделей данных и позволяющую совершать SQL запросы.
4. Интернет сервис – это служба, предназначенная для осуществления обмена данными по сети Интернет.

Уровень ядро является низшим уровнем доступа к оборудованию устройства и содержит такие компоненты, как файловая система, менеджер управления, сертификаты и т.д.

Наглядное представление архитектуры ОС iOS представлено на рисунке 2.

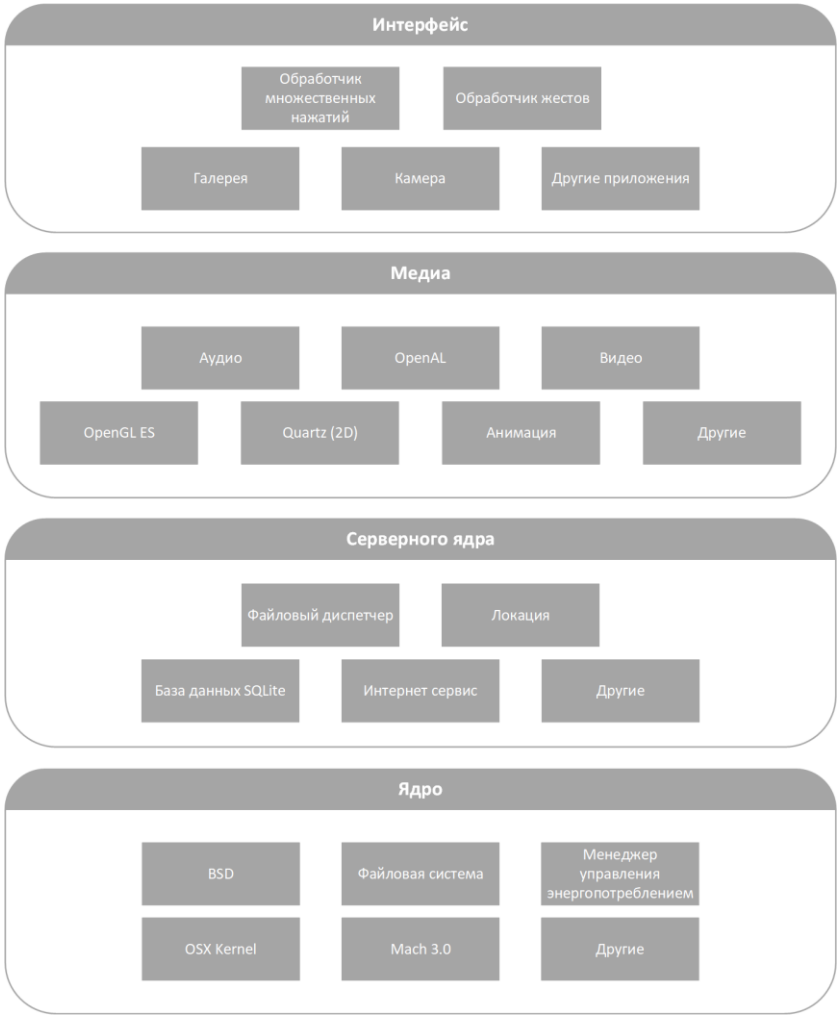


Рисунок 2 — схема архитектуры ОС iOS.

1.2. Анализ современных технологий кроссплатформенной разработки

При разработке современных мобильных ПО зачастую необходима поддержка наибольшего числа возможных мобильных ОС. Данная задача является сложновыполнимой, поскольку описанные ранее ОС Android и iOS, являющиеся наиболее популярными на мировом рынке, значительно отличаются по архитектуре, вследствие чего можно сделать вывод о различиях в технологиях создания мобильного ПО под каждую из них. По этой причине большинство разработчиков осуществляют разработку нескольких нативных приложений, каждое из которых рассчитано на конкретную ОС. Но данный процесс является очень трудозатратным и требует большого количества ресурсов. Для избежания значительных затрат на разработку подобных

приложений, была создана технология кроссплатформенного программирования для мобильных устройств.

Кроссплатформенность – это способность программного обеспечения работать с двумя и более аппаратными платформами, обеспечивающаяся благодаря использованию высокоуровневых языков программирования, сред разработки и выполнения, поддерживающих условную компиляцию, компоновку и выполнение кода для различных платформ.

Достоинствами данной технологии является:

1. экономия трудозатрат и ресурсов при создании приложения;
2. упрощенный процесс подготовки к разработке приложения;
3. возможность охватить большее количество пользователей;
4. реализация одного общего дизайна и функционала для разрабатываемого приложения.

Недостатки кроссплатформенного программирования зачастую заключаются в проблемной интеграции приложения и сложном процессе адаптации общего интерфейса под стилистические особенности каждой из платформ.

1.3. Выбор языка программирования

На начальном этапе разработки программного продукта необходимо выбрать язык программирования, на котором он будет реализован. Поскольку разрабатываемое приложение будет являться кроссплатформенным, требуется выбрать язык, поддерживающий данную технологию. Зачастую реализация данной технологии осуществляется с помощью создания дополнительных фреймворков. Примерами подобной реализации являются:

- React Native;
- Ionic;

- Xamarin;
- PhoneGap;
- Flutter.

React Native – это фреймворк, созданный корпорацией Facebook для реализации кроссплатформенных приложений с производительностью равной нативной. Языком программирования данного фреймворка является JavaScript с добавлением библиотеки React.

Главным достоинством данного инструмента является наличие функции Hot Reloading, предоставляющей возможность внесения правок в уже написанный код и добавление нового функционала в процессе выполнения программы.

Недостатком React Native является низкий уровень адаптивности некоторых компонентов фреймворка под различные платформы, из-за чего возникает необходимость в дополнительном изменении кода.

Фреймворк Ionic, предназначен для создания кроссплатформенно-гибридных приложений. Его главной особенностью является тесное взаимодействие с фреймворком Apache Cordova, предназначенным для преобразования веб-приложений в мобильные программы. Основными языками программирования для данного инструмента являются JavaScript, HTML и CSS.

Достоинством фреймворка является простота написания и тестирования кода, обусловленная тем, что все его компоненты автоматически адаптируются под каждую из платформ.

Главным недостатком Ionic является низкая производительность приложений, возникающая из-за отказа от нативных компонентов.

Xamarin – это фреймворк, предназначенный для создания кроссплатформенных мобильных приложений и базирующийся на C# и .NET.

В состав Xamarin входят два основных компонента Xamarin.Android/iOS и Xamarin.Forms.

Xamarin.Android и Xamarin.iOS позволяют приложениям иметь полный доступ к функционалу и интерфейсу, доступному для нативных решений, в результате чего приложения имеют высокую производительность.

Xamarin.Forms в отличие от предыдущего компонента является более простым в использовании, поскольку все его элементы полностью совместимы с любой платформой. Недостатком данного компонента является снижение производительности конечного приложения.

Подобно Ionic, фреймворк PhoneGap поддерживает применение веб-технологий при разработке приложения и также базируется на JavaScript, HTML и CSS. Но в отличие от Ionic, архитектура PhoneGap состоит из набора HTML-страниц, представленных в виде нативных компонентов.

Главным достоинством PhoneGap является то, что написанные на нём приложения заметно легче своих аналогов. Это достигается благодаря хранению всех страниц программы в локальном каталоге или облаке.

Недостатком данного фреймворка является низкая производительность и полная зависимость пользовательского интерфейса от конкретной ОС.

Flutter – это фреймворк, предназначенный для кроссплатформенной разработки и базирующийся на принципе создания приложений с единой кодовой базой. Создан данный фреймворк на базе языка Dart, разработанного компанией Google [3]. Основу Flutter составляет собственный графический движок, включающий обширный набор базовых виджетов [4].

Достоинствами данного фреймворка являются:

1. Высокий уровень производительности приложений, превосходящий аналоги, написанные на других фреймворках.
2. Поддержка возможности добавления нового кода во время работы приложения.

3. Большой набор базовых элементов, позволяющих создавать разнообразные интерфейсы.
4. Возможность запуска приложения под различными версиями Android и iOS без дополнительных настроек.

Недостаток Flutter заключается в отсутствии автоматической адаптации элементов интерфейса под стилистические особенности конкретной ОС.

На основании написанного ранее, можно сделать вывод о превосходстве фреймворка Flutter над его аналогами, на основании чего этот инструмент применяется в данной работе.

Вследствие выбора Flutter, как основного инструмента реализации кроссплатформенной разработки, основным языком программирования, используемым в представленной работе, является Dart.

1.4. Выбор среды разработки

При проектировании программного продукта важным является выбор среды разработки ПО. Выбор системы осуществляется на основании определённых заранее требований. В данной работе основополагающими параметрами являются поддержка языка программирования Dart и его фреймворка Flutter [5]. Инструментами разработки, удовлетворяющими данное условие, являются:

- IntelliJ IDEA;
- Visual Studio Code (VSCode);
- Android Studio.

IntelliJ IDEA – это интегрированная среда разработки, созданная компанией JetBrains. Данная среда предоставляет полный набор функций, интеграции с наиболее популярными технологиями и фреймворками. Основным направлением IntelliJ IDEA является поддержка языка

программирования Java, но несмотря на это, среда осуществляет поддержку и большого спектра других популярных языков.

Преимуществами данной системы являются:

1. поддержка большого количества языков;
2. возможность интегрирования систем контроля версий;
3. возможность настройки необходимого окружения;
4. быстрый анализ кода;
5. эргономичность.

Недостатком IntelliJ IDEA является требовательность к использованию ресурсов оперативной памяти компьютера.

Среда программирования Visual Studio Code была создана компанией Microsoft. Данная система представляет собой кроссплатформенный редактор кода для разработки приложений. Его главной особенностью является небольшой объем памяти, занимаемый системой.

Достоинством Visual Studio Code является возможность разработки программ на многих языках программирования.

Главным недостатком данной среды разработки является низкий уровень производительности.

Android Studio – это среда разработки, созданная компанией Google и предназначенная для работы с платформой Android. Основой данной системы является вышеупомянутая среда разработки IntelliJ IDEA, но при этом функционал Android Studio был расширен дополнительным инструментами, упрощающими процесс создания мобильных приложений.

Преимуществами данной среды программирования являются:

1. поддержка большого количества языков;
2. наличие удобного XML редактора;
3. поддержка системы контроля версий;

4. возможность эмуляции различных мобильных устройств;
5. возможность проведения тестирования и анализа кода;
6. возможность настройки окружения;
7. поддержка рендера средствами GPU.

Недостатком Android Studio является необходимость установки большого пакета инструментов для работы со средой.

На основание описанных ранее особенностей каждой из сред разработки, в качестве среды программирования, используемого в данной работе, была выбрана Android Studio.

1.5. Выбор дополнительных инструментов разработки

Выбранный в данной работе фреймворк Flutter и язык программирования Dart предоставляют большой набор базовых плагинов, позволяющих ускорить и упростить процесс разработки мобильного ПО за счёт использования готовых программных решений. Плагины содержат дополнительный код и данные, которые предоставляют автономные программные сервисы, позволяющие осуществлять модульный обмен и модификацию кода.

Для решения поставленной в данной работе задачи были выбраны следующие плагины:

- shared_preferences – это плагин, предназначенный для кратковременного хранения данных, запись которых осуществляется асинхронно. Оболочкой данного плагина в Android является SharedPreference, а в iOS — NSUserDefaults.
- http является плагином, предоставляющим набор функций и классов и обеспечивающим доступ к ресурсам протокола HTTP. Данный инструмент может использоваться как в командной строке, так и в браузере.

- sqflite – это плагин, предоставляющий функционал для создания, редактирования, а также удаления информации, в описанной ранее БД, SQLite.
- Плагин scoped_model позволяет легко передавать модель данных из родительского виджета его потомкам. Также он восстанавливает все дочерние элементы, которые использует модель при её обновлении.
- image_picker – это библиотека, предназначенная для работы с галереей и камерой устройства. В данном инструменте предусмотрены классы, работающие с видеофайлами и изображениями.
- url_launcher – это плагин, позволяющий открыть URL ссылку на мобильной платформе. В состав данной библиотеки входят процедуры, позволяющие открывать URL ссылки, а также проверяющие её на доступность.

Все, описанные ранее, плагины, выбранные для данной работы, предоставляют необходимый для её выполнения функционал в упрощенном и удобном виде.

1.6. Вывод

В данной главе были описаны особенности мобильных ОС Android и iOS. Также были приведены преимущества кроссплатформенной разработки над нативной, на основании которых были выбраны вспомогательные инструменты проектирования.

В качестве языка программирования был выбран Dart, поскольку он является главным языком для наиболее эффективного и производительного фреймворка Flutter, поддерживающего кроссплатформенное программирование. Из большого количества вспомогательных плагинов данного фреймворка были выбраны инструменты необходимые для

выполнения поставленной задачи. Также в данном разделе были проанализированы среды программирования, позволяющие создавать мобильное ПО, и на основании описанных данных была выбрана система Android Studio.

2. Реализация

В данной работе представлена библиотека регистрации и идентификации пользователя. Основным назначением модуля является обработка, хранение и передача персональных данных, полученных от пользователя.

Процесс разработки данной библиотеки можно разделить на следующие этапы:

- разработка структуры проекта;
- разработка структуры БД;
- разработка REST API;
- разработка классов.

2.1. Описание структуры проекта

В основе библиотеки лежит клиент-серверная архитектура, при этом взаимодействие между мобильным клиентом и сервером осуществляется по интерфейсу REST API. Использование REST API позволяет получать и отправлять необходимые данные, а также осуществлять их подтверждение [6].

Необходимо обеспечить выполнение запросов к серверу в асинхронном режиме, таким образом, чтобы ожидание ответа не сказывалось на отзывчивости пользовательского интерфейса [7].

Архитектура приложения, использующего разрабатываемую в данной работе библиотеку, фактически состоит из трёх модулей:

1. Модуль регистрации – это элемент, осуществляющий отображение текущего содержимого локальной БД, а также обработку и преобразование. При поступлении сигналов от модуля работы с сетью он извлекает из локальной БД необходимую информацию и обновляет соответствующие визуальные элементы.

2. Локальная БД – это элемент, хранящий в себе персональные данные пользователя.
3. Модуль работы с REST API – это элемент, осуществляющий взаимодействие с сервером.

В состав разрабатываемой библиотеки входят только Модуль регистрации и Локальная БД, а Модуль работы с REST API должен быть разработан пользователем библиотеки и прописан в списке её параметров. При этом функционал, поддерживающий асинхронное выполнение запросов из модуля работы, с выводом на экран соответствующих элементов, предоставляет библиотека.

Наглядное описание структуры приложения, использующего созданный в данной работе модуль, представлено на рисунке 3.

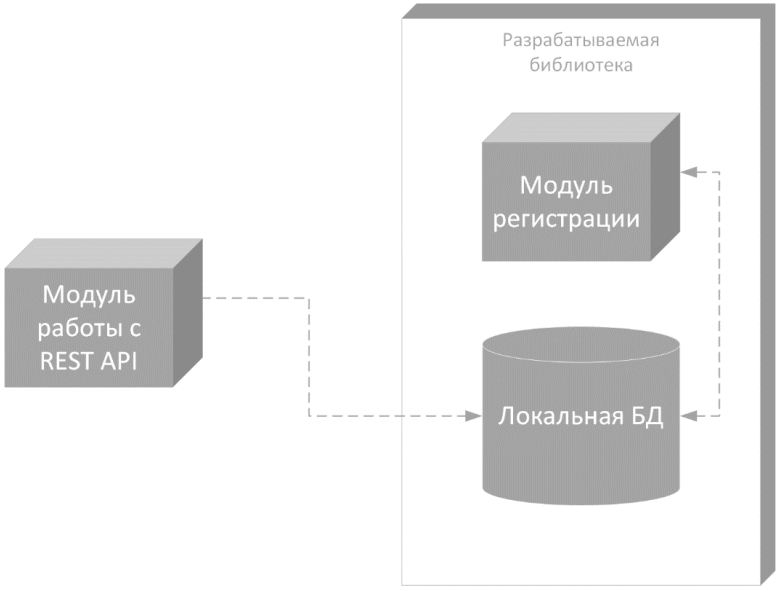


Рисунок 3 — диаграмма структуры приложения.

Модуль регистрации является сложным элементом, реализующим весь функционал данной библиотеки. Данный модуль состоит из 3 компонентов:

1. UI Screen – это компонент, содержащий в себе все элементы пользовательского интерфейса, а именно виджеты.
2. Компонент BLoC предназначен для управления состояниями виджетов, для уведомления о изменении которых, применяются

потоки. Данный уровень является промежуточным между данными и компонентами пользовательского интерфейса.

3. Repository – это компонент, отвечающий за выборку информации из одного или нескольких источников данных и обработку ее для классов пользовательского интерфейса.

Диаграмма взаимодействия между компонентами модуля регистрации представлена на рисунке 4.

Рисунок 4 — диаграмма взаимодействия компонентов модуля регистрации.

2.2. Структура БД

При разработке мобильного приложения для хранения необходимого набора данных применяется база данных SQLite. SQLite – это комплексная встроенная реляционная БД, представленная в виде библиотеки с которой взаимодействует приложение [8]. Для обмена информацией в библиотеке предусмотрены функции, что позволяет упростить код и уменьшает время, затрачиваемое на отклик системы.

SQLite осуществляет хранение данных в едином стандартном файле на выполняющем устройстве [9]. При этом простота достигается благодаря блокировке файла, хранящего данные, в начале выполнения процесса записи. Наглядное представление БД отображено на рисунке 5.

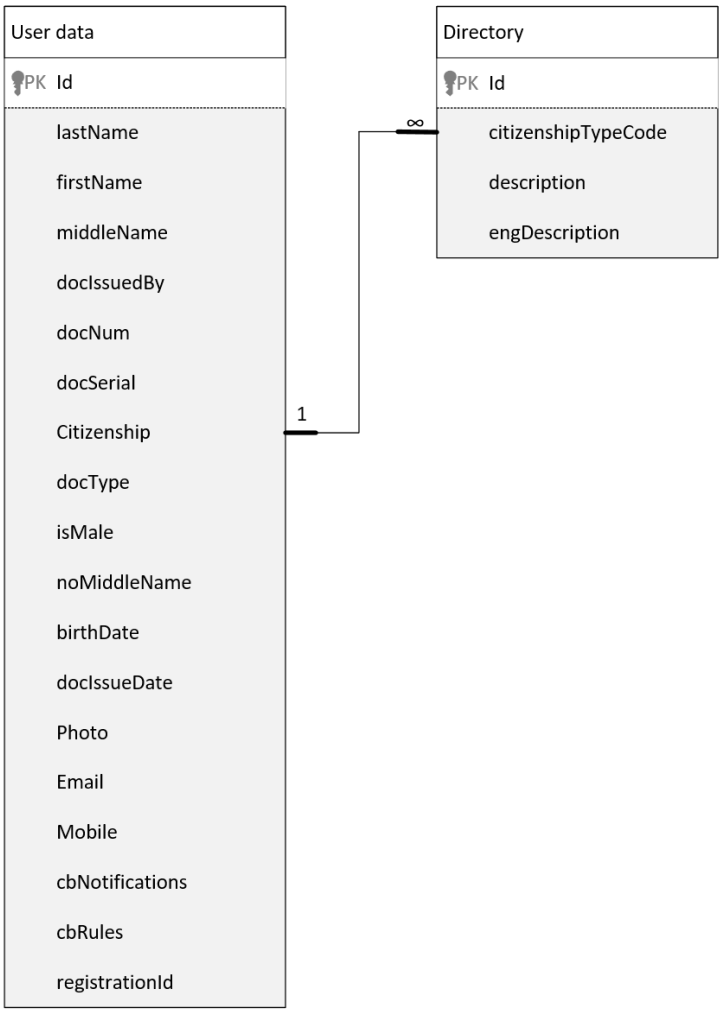


Рисунок 5 — диаграмма базы данных библиотеки.

Ниже представлены таблицы с описанием полей и структуры разработанной БД.

В Таблице 1 представлена структура таблицы «Данные пользователя».

Таблица 1 — Структура таблицы «Данные пользователя».

Название поля	**Описание поля**	**Тип данных**
Id	Идентификатор пользователя	Integer
lastName	Фамилия пользователя	Text
firstName	Имя пользователя	Text
middleName	Отчество пользователя	Text
docIssuedBy	Кем был выдан документ удостоверяющий личность пользователя	Text
docNum	Номер документа удостоверяющего личность пользователя	Text

Продолжение Таблицы 1.

Название поля	Описание поля	Тип данных
docSerial	Серия документа удостоверяющего личность пользователя	Text
Citizenship	Гражданство	Text
docType	Тип документа	Text
isMale	Пол пользователя	Integer
noMiddleName	Наличие у пользователя отчества	Integer
birthDate	Дата рождения пользователя	Text
docIssueDate	Дата выдачи документа удостоверяющего личность пользователя	Text
Photo	Аватар (фотография)	Blob
Email	E-майл пользователя	Text
Mobile	Телефон пользователя	Text
cbNotifications	Согласие на получение уведомлений от системы	Integer
cbRules	Согласие на обработку персональных данных пользователя	Integer
registrationId	Уникальный идентификатор зарегистрированного пользователя	Text

В Таблице 2 представлена структура таблицы «Справочник».

Таблица 2 — Структура таблицы «Справочник».

Название поля	Описание поля	Тип данных
Id	Идентификатор справочника	Integer
citizenshipTypeCode	Код страны	Text
Description	Название страны на русском языке	Text
engDescription	Название страны на английском языке	Text

2.3. Описание REST API

Разработка API является неотъемлемой частью мобильного приложения, базирующегося на клиент-серверной архитектуре [10]. В рамках данной работы была разработана библиотека регистрации и идентификации пользователя. Для демонстрации работоспособности данного модуля был создан API, соответствующий архитектурному стилю *REST* и поддерживающий следующий функционал:

- получение справочника;
- отправка одноразового кода регистрации;

- регистрация нового пользователя.

В качестве примера реутилизации запроса, ниже будет рассмотрен запрос на регистрацию нового пользователя.

При регистрации пользователя, по спецификации REST API, необходимо использовать метод POST протокола HTTP [11]. Для осуществления данного запроса, необходимо сформировать URL со следующими элементами: название микросервиса/название сущности/операция. В тело запроса входят поля, необходимые для регистрации пользователя.

В случае благополучного завершения регистрации пользователя возвращается ответ с кодом 200 и объектом, содержащим идентификатор пользователя. В случае отказа пользователю в регистрации возвращается ответ с кодом 200 и объектом, содержащим сообщение об ошибке. В ином случае возвращается код 400 и сообщение об ошибке.

Листинг 1. Запрос на регистрацию пользователя:

```
Метод: POST

URI:https://68f5f049-374a-4665-8bbf-898124a6af24.mock.pstmn.io/user/new

Тело запроса:
        {
          "lastName": "String",
          "firstName": "String",
          "middleName": "String",
          "birthDate": "String",
          "citizenshipCode": "String",
          "email": "String",
          "otpRqId": int,
          "otp": "String",
          "idType": int,
          "idNumber": "String",
```

```
            "idIssuer": "String",
            "idIssueDate": "String",
            "photo": "String",
            " gender ": int
          }
Ответ:
      200   OK
            {
              "resultCode": int,
              "desc": "String",
              "registrationId": int
            }
      400   Bad Request
            {
             "desc": "String"
            }
```

Полное описание *REST API* представлено в приложении В.

2.4. Описание классов

При реализации библиотеки регистрации и идентификации пользователя были созданы классы, предназначенные для хранения различных типов данных [12]. Наглядное взаимодействие данных классов представлено на рисунке 6.

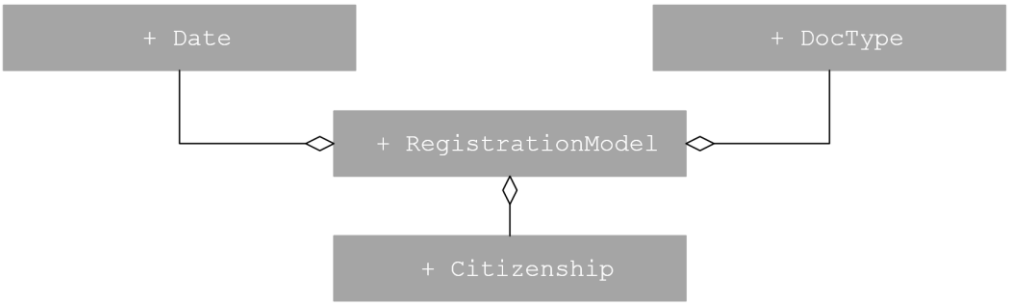


Рисунок 6 — диаграмма классов.

Класс «RegistationModel» предназначен для хранения персональных данных пользователей, а также для осуществления обработки вводимых пользователем данных. Структура класса «RegistationModel» представлена на рисунке 7.

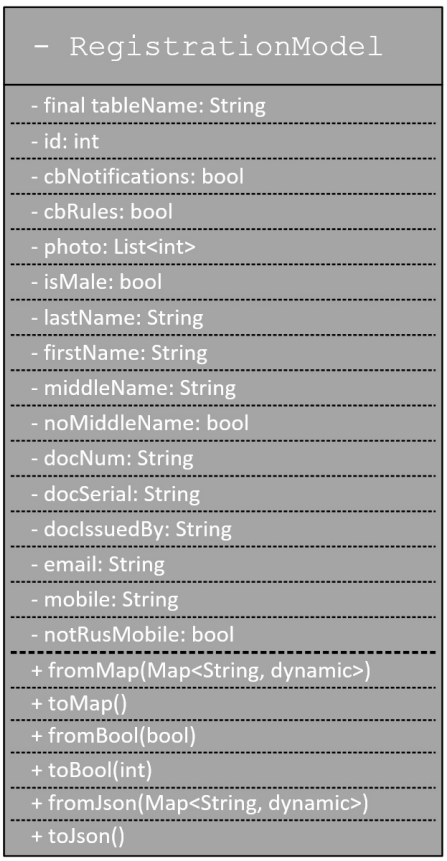


Рисунок 7 — схема структуры класса «RegistationModel».

«DocType» является классом, предназначенным для хранения, обработки и преобразования информации о типах документов, удостоверяющих личность пользователя. Наглядное отображение структуры класса «DocType» представлена на рисунке 8.

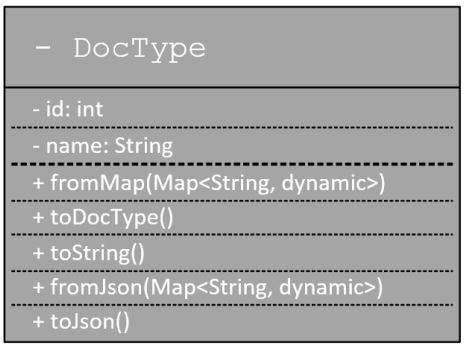


Рисунок 8 — схема структуры класса «DocType».

Класс «Citizenship» предназначен для хранения информации о странах, также содержит методы, позволяющие осуществлять преобразовать объекта

класса к необходимому типу. Структура класса «Citizenship» представлена на рисунке 9.

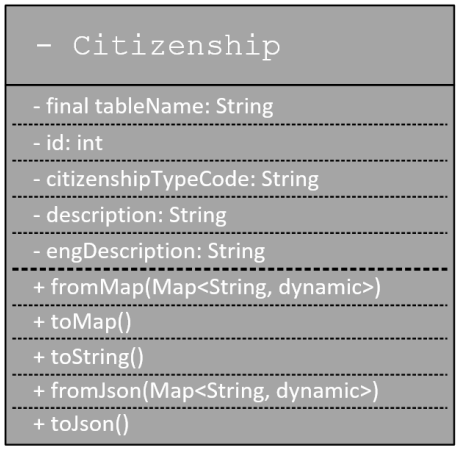


Рисунок 9 — схема структуры класса «Citizenship».

«Date» — это класс, позволяющий хранить необходимые даты, а также осуществлять с ними арифметические операции и преобразовывать к строковому типу. Структура класса «Data» отображена на рисунке 8.

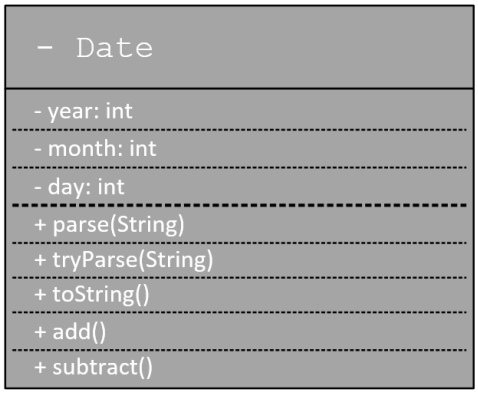


Рисунок 10 — схема структуры класса «Date».

2.5. Реализация программы

Приложение, разработанное для демонстрации работоспособности библиотеки регистрации и идентификации пользователя, обладает обширным интерфейсом, в состав которого входя стандартные страницы мобильного приложения, всплывающие окна, а также встроенные в приложение страницы

web-браузера. Наглядная схема взаимодействия элементов пользовательского интерфейса представлена на рисунке 11.

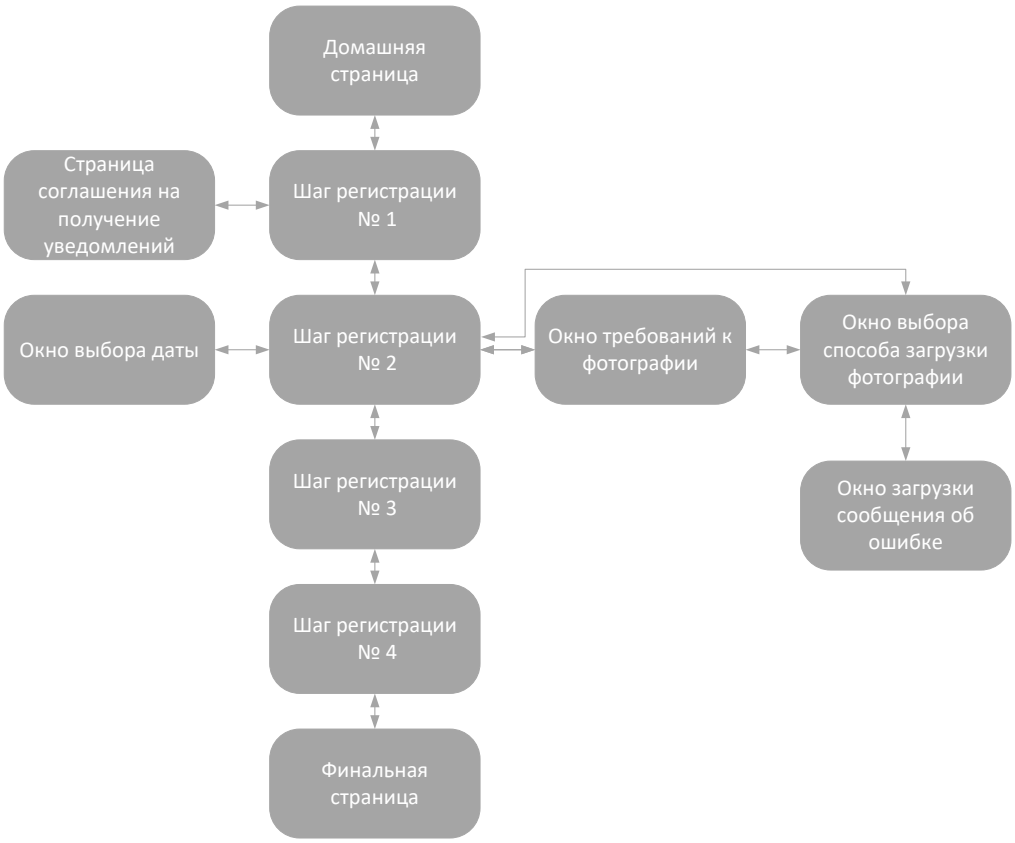


Рисунок 11 — схема пользовательского интерфейса.

В интерфейс, разрабатываемый в библиотеке, входит функционал по работе с изображениями. Данный функционал осуществляет получение фотографии от приложения «Камера» или из файловой системы, после чего осуществляется передача полученного изображения в окно редактирования. После завершения редактирования, полученное изображение сохраняется и передаётся на проверку стороннему сервису. При успешном завершении проверки, изображение загружается на экран пользователя. В случае несоответствия фотографии поставленным критериям, изображение удаляется из хранящей его переменной. Подобный же алгоритм выполняется и в случае отказа стороннего сервиса.

Наглядно алгоритм выполнения функционала по обработке изображения представлен на рисунке 12.

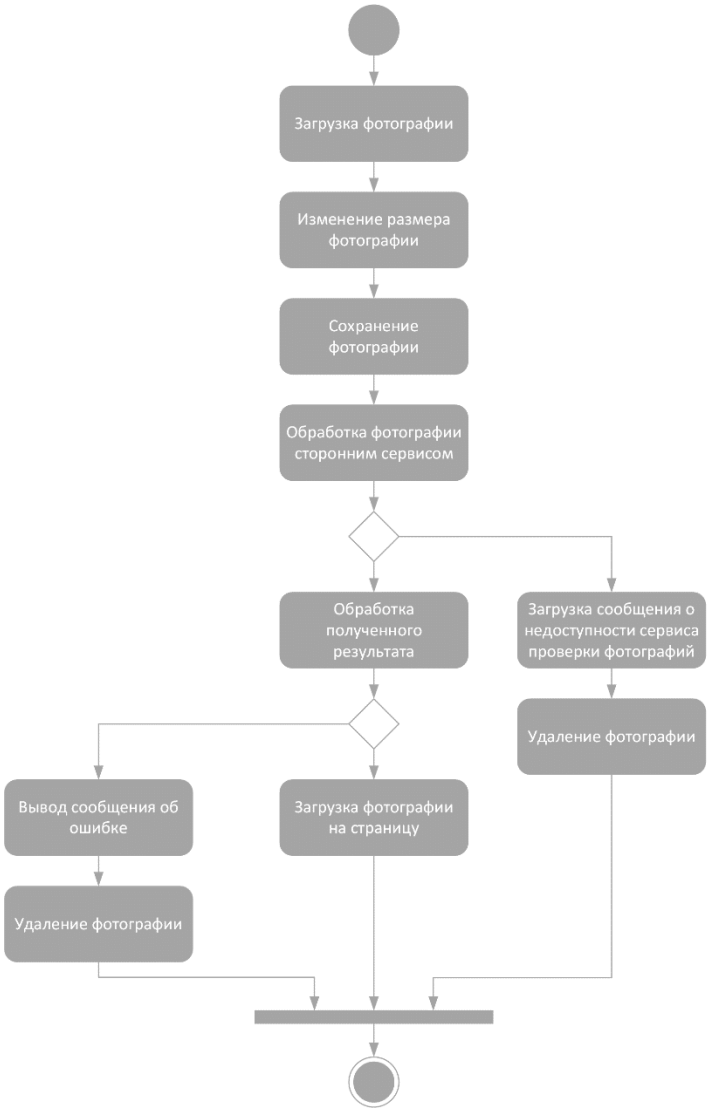


Рисунок 12 — диаграмма активности обработки изображения.

2.6. Вывод

В данном разделе описывается процесс реализации библиотеки регистрации и идентификации пользователя, а также приложения демонстрирующего её работу.

В процессе создания библиотеки и приложения была спроектирована структура проекта, а также были разработаны БД, REST API и классы, необходимые для обработки и хранения данных.

3. Инструкция пользователя

Приложение «Authorization library» является примером применения разрабатываемого в данной работе универсального модуля регистрации пользователя, позволяющего разработчику мобильного ПО уменьшить трудозатраты, благодаря использованию готового инструмента.

3.1. Описание пользовательского интерфейса

Приложение представляет собой несколько экранов:

1. Экран входа. Экран представляет название приложения, логотип и кнопку начала регистрации.

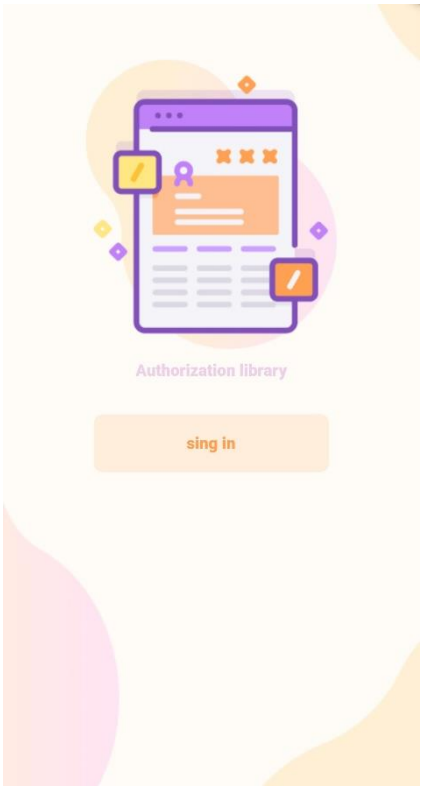


Рисунок 13 — экран входа.

2. Экран разрешения на обработку персональных данных. Экран содержит два чек-бокса, активируя которые пользователь даёт соглашение на получение уведомлений по электронной почте и SMS, на обработку персональных данных. Также экран содержит ссылку

«Соглашение на получение уведомлений по электронной почте и SMS».

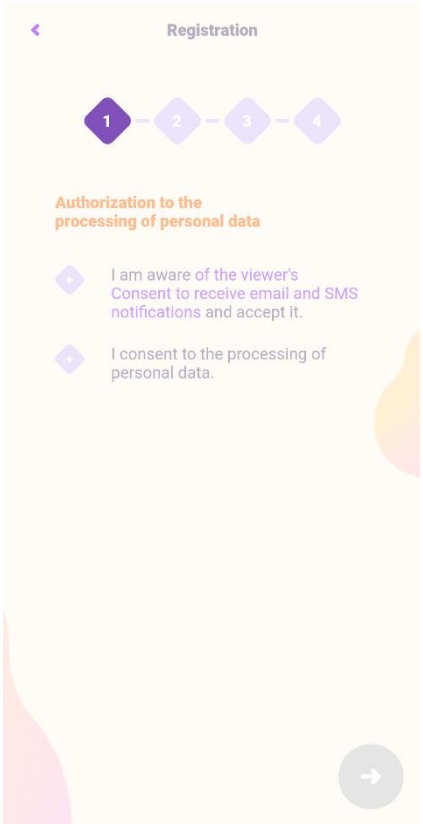


Рисунок 14 — экран разрешения на обработку персональных данных.

3. Экран ввода персональных данных. На данном экране пользователю предоставляется возможность ввести или выбрать следующие персональные данные:
   1) фотография;
   2) пол;
   3) фамилия;
   4) имя;
   5) отчество (при наличии);
   6) дата рождения;
   7) гражданство;
   8) тип документа, подтверждающего личность;
   9) серия и номер документа;

10) кем выдан документ;

11) когда выдан документ.

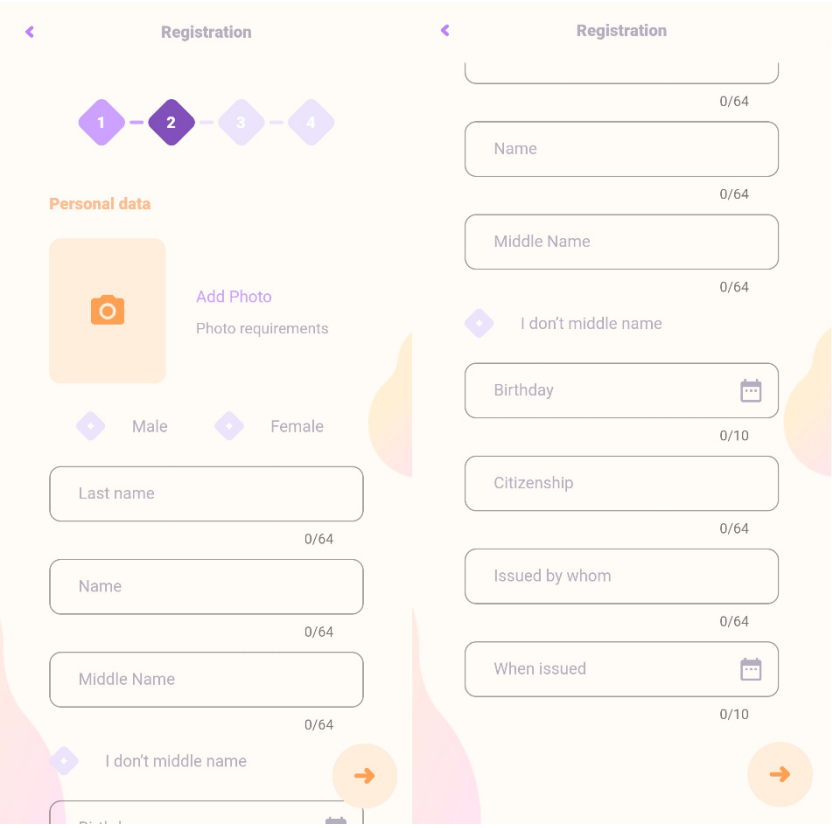


Рисунок 15 — экран ввода персональных данных.

4. Всплывающее окно требований к фотографии. В окне отображается информация о требованиях к загружаемому изображению. Также в нижней части окна расположена кнопка добавления фотографии и кнопка закрытия окна.

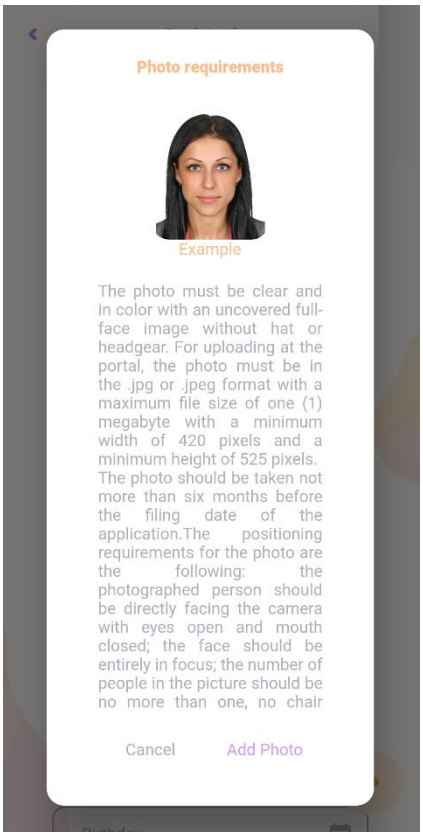


Рисунок 16 — всплывающее окно требований к фотографии.

5. Всплывающее окно добавления фотографии. В окне отображается кнопка создания фотографии, кнопка выбора фотографии и кнопка закрытия окна.

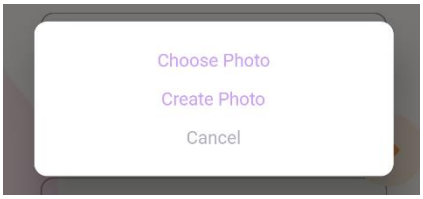


Рисунок 17 — всплывающее окно добавления фотографии.

6. Всплывающее окно добавления фотографии. В окне отображается список параметров, которым не удовлетворяет загруженная фотография. Также в нижней части окна расположена кнопка добавления фотографии и кнопка закрытия окна.

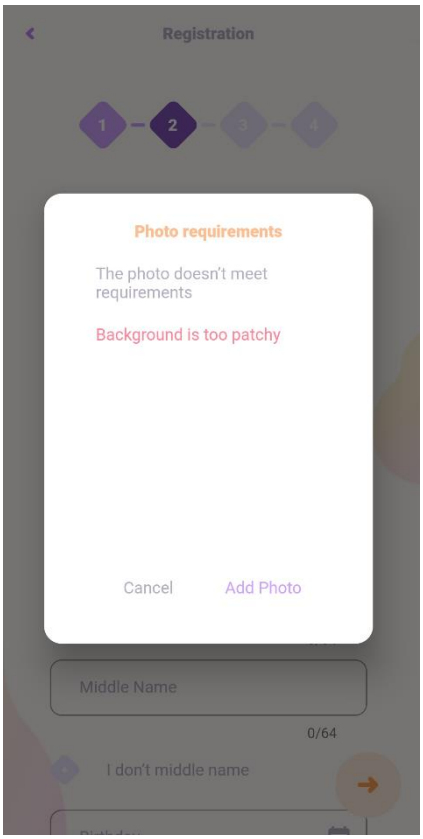


Рисунок 18 — всплывающее окно со списком параметров, не удовлетворяемых фотографией.

7. Экран ввода контактных данных. Экран, предназначенный для ввода контактных данных (e-mail и номер телефона). На странице отображается кнопка оправки кода подтверждения.

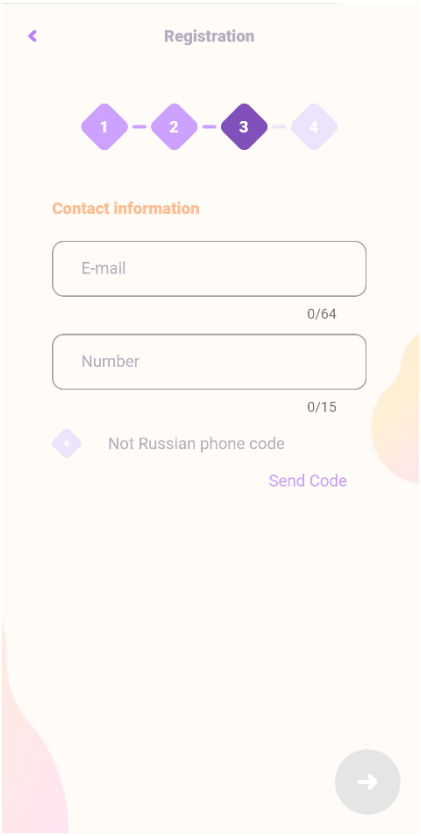


Рисунок 19 — экран ввода контактных данных.

8. Экран ввода контактных данных, после нажатия кнопки “Send code”. На странице добавляется сообщение об отправке кода подтверждения и поле для его ввода.

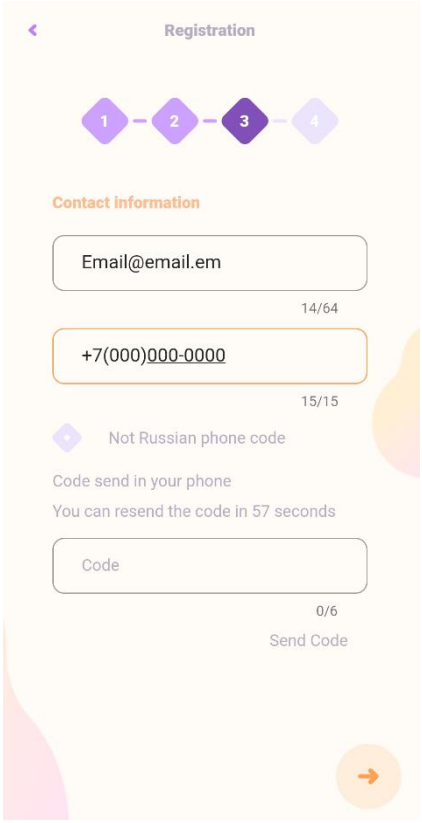


Рисунок 20 — экран ввода контактных данных, после нажатия кнопки “Send code”.

9. Экран вывода полученных данных. Экран предназначен для вывода всех введённых пользователем персональных данных.

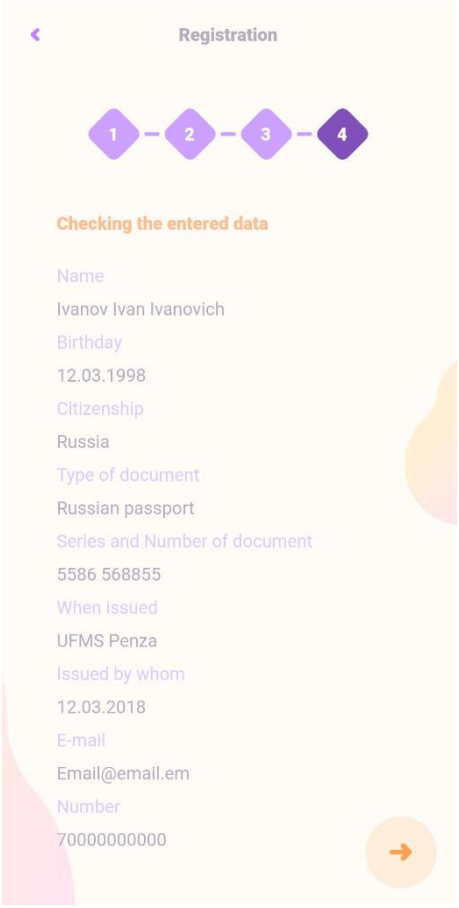


Рисунок 21 — экран вывода полученных данных.

3.2. Вывод

Описанный в данной главе интерфейс позволяет осуществлять ввод персональных данных клиента. В нём предусмотрена возможность возникновения внештатных ситуаций, в случае которых пользователю в корректной форме выводится сообщение об ошибке и предоставляется возможность повторного использования функционала.

Данный интерфейс является простым и интуитивно понятным для пользователя.

4. Охрана труда

Появление современных компьютеров произвело техническую революцию во всех сфера жизни человека. Но не смотря на огромный вклад, внесённый данными устройствами, нельзя отрицать и негативных сторон данного технического прогресса. Главный недостаток современных компьютеров заключается в их негативном влиянии на здоровье человека. Для минимизации негативного влияния компьютера были разработаны требования по безопасности, при работе с ПВЭМ.

В состав данных требований входят следующие пункты:

- ПЭВМ должны соответствовать требованиям настоящих санитарных правил, и каждый их тип подлежит санитарно-эпидемиологической экспертизе с оценкой в испытательных лабораториях, аккредитованных в установленном порядке;
- допустимые уровни звукового давления и уровней звука, создаваемых ПЭВМ, не должны превышать установленных норм;
- временные допустимые уровни электромагнитных полей, создаваемых ПЭВМ, не должны превышать установленных норм;
- концентрации вредных веществ, выделяемых ПЭВМ в воздух помещений, не должны превышать предельно допустимых концентраций, установленных для атмосферного воздуха;
- мощность экспозиционной дозы мягкого рентгеновского излучения в любой точке на расстоянии 0,05 м от экрана и корпуса ВДТ (на электронно-лучевой трубке) при любых положениях регулировочных устройств не должна превышать 1 мкЗв/час (100 мкР/час);
- документация на проектирование, изготовление и эксплуатацию ПЭВМ не должна противоречить требованиям настоящих Санитарных правил.

Также необходимо чтобы используемое ПЭВМ, поддерживало возможность поворота корпуса и его фиксации в установленном положении. Расцветка ПЭВМ должна состоять из спокойных цветов, не отвлекающих внимание.

При организации условий труда также следует учитывать, требования к освещённости рабочего места сотрудника:

- помещения, предназначенные для размещения рабочих мест пользователей персональных компьютеров, должны иметь естественное и искусственное освещение;
- естественное освещение помещений, предназначенных для размещения рабочих мест, должно осуществляться через световые проемы, ориентированные преимущественно на север и северо-восток;
- в случае иной ориентации световых проемов, необходимо предусматривать эффективные средства регулирования интенсивности естественного освещения;
- оконные проемы в помещениях, предназначенных для использования компьютеров, необходимо оборудовать устройствами регулирования интенсивности естественного освещения типа жалюзи, занавесей, внешних козырьков и др.

Уровень шума в рабочем помещении, также регулируется санитарными нормами. Зачастую главным источником шума в рабочем помещение является офисное оборудование. Для снижения подобного шума необходимо предусмотреть специализированные амортизирующие прокладки, предусмотренные нормативными документами. Уровень шума на рабочих местах не должен превышать 50 дБА.

Соблюдение описанных ранее мер является обязательным требованием для любого рабочего места, так как оно позволяет снизить уровень негативного влияния на здоровье человека при работе с ПЭВМ.

5. Заключение

В ходе данной работы была реализована библиотека регистрации и идентификации пользователя для мобильного клиента, и были приобретены навыки работы с языком программирования Dart и его фреймворком Flutter. При реализации поставленной задачи были получены следующие результаты:

- проведён анализ современных мобильных операционных систем, а также современных технологий кроссплатформенной разработки мобильного ПО;
- разработана структура базы данных для мобильного клиента, позволяющая хранить введённую пользователем информацию;
- разработан интерфейс библиотеки регистрации и идентификации пользователя, предназначенный для взаимодействия с пользователем;
- разработаны асинхронные механизмы обработки, позволяющие осуществлять взаимодействие со сторонним сервисом;
- разработано и отлажено мобильное приложение и интерфейс REST API, демонстрирующие работоспособность созданной библиотеки регистрации и идентификации пользователя.

Библиотека корректно интегрируется в разрабатываемые приложения.

Список используемых источников

1. Таненбаум Эндрю, Бос Х. Современные операционные системы. — СПб.: «Питер», 2018. — С. 637-810.
2. Медникс З. Программирование под Android. 2-е изд. — СПб.: «Питер», 2013. — С. 345-475.
3. Фрэнк Заметти. Flutter на практике — Перевод с английского. — М.: «ДМК Пресс», 2020. — С. 127-235.
4. Андрей Алеев. Быстрый старт Flutter-разработчика. — Екб.: «Издательские решения», 2019. — С. 73-102.
5. Крис Баккет. Dart в действии — Перевод с английского. — М.: «ДМК Пресс», 2016. — С. 135-309.
6. Гамма Э., Джонсон Р., Хелм Р. Паттерны объектно-ориентированного проектирования. — СПб.: «Питер», 2020. — С. 245-307.
7. Гамма Э., Влиссидес Дж., Джонсон Р., Хелм Р. Приемы объектно-ориентированного проектирования. Паттерны проектирования — СПб.: «Питер», 2019. — С. 267-312.
8. Осипов Д. Л. Технологии проектирования баз данных. — М.: «ДМК Пресс», 2019. — С. 135-215.
9. Новиков Б. А., Горшкова Е. А. Основы технологий баз данных. Учебное пособие — М.: «ДМК Пресс», 2019. — С. 165-233.
10. Д. Куроуз, К. Росс. Компьютерные сети. Нисходящий подход. — М.: Издательство «Э», 2016. – С. 912.
11. В. Олифер, Н. Олифер. Компьютерные сети. Принципы, технологии, протоколы. Учебник. — СПб.: «Питер», 2016. — С. 256-307.
12. Р. Мартин. Чистый код. Создание, анализ и рефакторинг. — СПб.: «Питер», 2013. — С. 164-181.

Приложение А. Листинги программы на Dart

Файл «main.dart»

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart' ;
import 'package:reg_model/views/home_screen.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget{
  @override
  Widget build(BuildContext context) {
    SystemChrome.setPreferredOrientations([
      DeviceOrientation.portraitUp,
    ]);
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}
```

Файл «dict_page.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/views/elements/proces_request.dart';
import 'package:reg_model/components/network/dict.dart';
import 'package:reg_model/components/model/registration.dart';
import 'package:reg_model/components/bd.dart';
import 'package:reg_model/views/reg2_page.dart';
import 'package:reg_model/views/elements/error_snack_bar.dart';

class DictPage extends StatefulWidget {
  final _formKey;
  RegistrationModel model;
  DictPage(this._formKey ,this.model);

  @override
  _DictPageState createState() => _DictPageState();
}

class _DictPageState extends State<DictPage> {
  @override
```

```
  void initState() {
    super.initState();
    DatabaseHelper helper = DatabaseHelper.instance;
    helper.upsertReg(widget.model).then((model){
      Dict().fetchDict().then((success) {
        if(success.resultCode == 0){
          Dict().saveResponseData(success);
          Navigator.pop(context);
          Route route = MaterialPageRoute(
            builder: (context) => RegistrationPage2(),
          );
          Navigator.push(context, route);
        } else{
          Navigator.pop(context);
          ErrorSnackBar(widget._formKey, 'Service error, check
your Internet connection and try again');
        }
      }).catchError((e) {
        print(e.toString());
        Navigator.pop(context);
        ErrorSnackBar(widget._formKey, e.toString());
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return ProcesRequest();
  }
}
```

Файл «finish_screen.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/views/elements/title_page.dart';

class FinishPage extends StatefulWidget {
  FinishPage({Key key,}) : super(key: key);
  @override
  _FinishPageState createState() => _FinishPageState();
}

class _FinishPageState extends State<FinishPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        fit: StackFit.expand,
        children: <Widget>[
```

```
          DecoratedBox(
            position: DecorationPosition.background,
            decoration: BoxDecoration(
                image: DecorationImage(
                  image:
AssetImage('assets/images/finish_screen_background.png'),
                  fit: BoxFit.cover,
                )
            ),
          ),
          Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Center(
                child: Column(
                  children: <Widget>[
                    Image.asset('assets/images/finish_icon.png',
                      width: 250,
                      height: 250,
                    ),
                    TitlePage('The application is registered!',
EdgeInsets.only(bottom: 100))
                  ],
                ),
              ),
            ],
          )
        ],
      ),
    );
  }
}
```

Файл «home_screen.java»

```
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:reg_model/views/reg1_page.dart';
import 'package:reg_model/step.dart';

class MyHomePage extends StatefulWidget {
  MyHomePage({Key key,}) : super(key: key);
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  List<String> items = ['ru','en'];
  String language = 'ru';
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        fit: StackFit.expand,
        children: <Widget>[
          DecoratedBox(
            position: DecorationPosition.background,
            decoration: BoxDecoration(
                image: DecorationImage(
                  image:
AssetImage('assets/images/home_screen_background.png'),
                  fit: BoxFit.cover,
                )
            ),
          ),
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Padding(
                padding: EdgeInsets.only(left:40, top:50,
bottom: 50),
                child: Container(
                  padding: EdgeInsets.symmetric(horizontal: 10,
vertical: 2),
                  child: Container()
                ),
              ),
              Center(
                child: Column(
                  children: <Widget>[
                    Image.asset('assets/images/icon.png',
                      width: 250,
                      height: 250,
                    ),
                    Container(height: 10),
                    Text('Authorization library',
                      style: TextStyle(
                          color: MyColor().libraryNameColor,
                          fontFamily: 'Roboto-Black',
                          fontSize: 14
                      ),
                    ),
                    Padding(
                      padding: EdgeInsets.only(top: 30),
                      child: FlatButton(
                        shape: new RoundedRectangleBorder(
                          borderRadius: new
BorderRadius.circular(8),
```

```
                        ),
                        color: MyColor().activeButtonColor,
                        padding: EdgeInsets.symmetric(vertical:
17, horizontal: 80),
                        child: Text('sing in',
                            style: TextStyle(
                                color: MyColor().buttonColor,
                                fontFamily: 'Roboto-Black',
                                fontSize: 14
                            )
                        ),
                        onPressed: (){
                          Route route =MaterialPageRoute(
                              builder: (context) =>
RegistrationPage1()
                          );
                          Navigator.push(context, route);
                        },
                      ),
                    ),
                  ],
                ),
              ),
            ],
          )
        ],
      ),
    );
  }
}
```

Файл «new_page.dart»

```
import 'dart:core';
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:reg_model/views/elements/proces_request.dart';
import 'package:reg_model/components/network/new.dart';
import 'package:reg_model/components/model/registration.dart';
import 'package:reg_model/components/bd.dart';
import 'package:reg_model/views/reg4_page.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/views/elements/error_snack_bar.dart';

class NewPage extends StatefulWidget {
  final _formKey;
  RegistrationModel model;
  int otpRqId;
  String code;
```

```
    NewPage(this._formKey, this.model, this.otpRqId, this.code);

    @override
    _NewPageState createState() => _NewPageState();
  }

  class _NewPageState extends State<NewPage> {
    @override
    void initState() {
      super.initState();
      var _new= new New(
          widget.model.lastName,
          widget.model.firstName,
          widget.model.noMiddleName == false ? "" :
  widget.model.middleName ?? "",
          toString(widget.model.birthDate),
          widget.model.citizenship.citizenshipTypeCode,
          widget.model.email,
          widget.otpRqId,
          widget.code,
          widget.model.docType.id,
          (widget.model.docSerial ?? '') + widget.model.docNum,
          widget.model.docIssuedBy,
          toString(widget.model.docIssueDate),
          base64.encode(widget.model.photo),
          widget.model.isMale == true ? 0 : 1
      );

      DatabaseHelper helper = DatabaseHelper.instance;
      helper.upsertReg(widget.model).then((model){
        String error;
        _new.fetchNew().then((success) {
          if(success.resultCode == 0){
            _new.saveResponseData(success);
            Navigator.pop(context);
            Route route = MaterialPageRoute(
              builder: (context) => RegistrationPage4(),
            );
            Navigator.push(context, route);
            return;
          } else if (success.resultCode == 1) {
            error = success.desc;
          } else {
            error = 'Service error, check your Internet connection
  and try again';
          }
          Navigator.pop(context);
          ErrorSnackBar(widget._formKey, error);
        }).catchError((e) {
          print(e.toString());
```

```
        error = e.toString();
        Navigator.pop(context);
        ErrorSnackBar(widget._formKey, error);
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return ProcesRequest();
  }
}
```

Файл «reg1_page.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/views/elements/register.dart';
import 'package:reg_model/views/elements/check_box.dart';
import 'package:reg_model/components/model/registration.dart';
import 'package:reg_model/views/elements/title_page.dart';
import 'package:reg_model/views/dict_page.dart';

class RegistrationPage1 extends StatefulWidget {
  RegistrationPage1({Key key,}) : super(key: key);

  @override
  _RegistrationPage1State createState() =>
_RegistrationPage1State();
}

class _RegistrationPage1State extends State<RegistrationPage1>{
  final _formKey = GlobalKey<FormState>();
  RegistrationModel _model = RegistrationModel();
  bool processRequest = false;

  @override
  Widget build(BuildContext context) {
     return RegistrationPage(
         Form(
             key: _formKey,
             child: Column(
               crossAxisAlignment: CrossAxisAlignment.stretch,
               children: <Widget> [
                 TitlePage(
                   'Authorization to the\nprocessing of personal
data',
                   EdgeInsets.only(bottom: 30,top: 12),
                 ),
                 CheckBox(
                   value: _model.cbNotifications,
```

```
                    onChanged: (){
                      setState(() {
                        _model.cbNotifications =
!_model.cbNotifications;
                      });
                    },
                    text: 'I am aware [of the viewer\'s\nConsent
to receive email and SMS
\nnotifications[https://www.garant.ru/products/ipo/prime/doc/704
78098/#200]] and accept it.',
                    crossAxisAlignment: CrossAxisAlignment.start,
                    iconAlignment: Alignment.topLeft,
                  ),
                  Container(
                    padding: EdgeInsets.only(top: 20),
                    child: CheckBox(
                      value: _model.cbRules,
                      onChanged: (){
                        setState(() {
                          _model.cbRules = !_model.cbRules;
                        });
                      },
                      text: 'I consent to the processing
of\npersonal data.',
                      crossAxisAlignment:
CrossAxisAlignment.start,
                      iconAlignment: Alignment.topLeft,
                    ),
                  )
                ],
              )
          ),
        (){
          if(_model.cbRules && _model.cbNotifications){
            Route route = MaterialPageRoute(
                builder: (context) => DictPage(_formKey, _model)
            );
            Navigator.push(context, route);
          }
        },
        1,
        activeButton: _model.cbRules && _model.cbNotifications
      );
  }
}
```

Файл «reg2-page.dart»

```
import 'package:flutter/material.dart';
import 'package:flutter_masked_text/flutter_masked_text.dart';
```

```
import 'package:flutter_typeahead/flutter_typeahead.dart';
import 'package:reg_model/views/elements/register.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/views/reg3_page.dart';
import 'package:reg_model/components/docType.dart';
import
'package:reg_model/components/citizenship_stream_builder.dart';
import 'package:reg_model/views/elements/check_box.dart';
import 'package:reg_model/views/elements/photo_page.dart';
import 'package:reg_model/components/model/registration.dart';
import 'package:reg_model/views/elements/date_text_field.dart';
import 'package:reg_model/views/elements/series_number.dart';
import
'package:reg_model/views/elements/decorated_text_filed.dart';
import 'package:reg_model/views/elements/title_page.dart';
import 'package:reg_model/components/validator.dart';
import 'package:reg_model/views/elements/error_snack_bar.dart';
import 'package:reg_model/views/update_page.dart';
import 'package:reg_model/components/bd.dart';

class RegistrationPage2 extends StatefulWidget {
  RegistrationPage2({
    Key key,
  }) : super(key: key);

  @override
  _RegistrationPage2State createState() =>
_RegistrationPage2State();
}

class _RegistrationPage2State extends State<RegistrationPage2> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController lastNameTextController =
  new TextEditingController();
  final TextEditingController firstNameTextController =
  new TextEditingController();
  final TextEditingController middleNameTextController =
  new TextEditingController();
  final MaskedTextController birthDateTextController =
  new MaskedTextController(mask: '00.00.0000');
  final TextEditingController citizenshipTextController =
  new TextEditingController();
  final TextEditingController docSerialTextController =
  new TextEditingController();
  final TextEditingController docNumberTextController =
  new TextEditingController();
  final TextEditingController docIssuedByTextController =
  new TextEditingController();
  final MaskedTextController docIssueDateTextController =
  new MaskedTextController(mask: '00.00.0000');
```

```
  RegistrationModel _model = RegistrationModel();
  final CitizenshipStreamBuilder citizenshipList = new
CitizenshipStreamBuilder();

  bool autovalid = false;
  ///убрать данную функцию
  update() {
    setState(() {});
    if(autovalid)
      _formKey.currentState.validate();
  }

  DateTime minDateIssue(String value){
    if(value == "" || value.length != 10)
      return DateTime(1900);
    DateTime date = toDateTime(value);
    if(_model.docType != null){
      if(_model.docType.id == 21){
        return DateTime(date.year + 14, date.month, date.day);
      }
    }
    return DateTime(date.year, date.month, date.day);
  }

  DateTime maxDateBirth(String value){
    if(_model.docType != null){
      if(_model.docType.id == 21){
        return DateTime(DateTime.now().year - 14,
DateTime.now().month, DateTime.now().day);
      }
    }
    return docIssueDateTextController.text == "" ||
docIssueDateTextController.text.length != 10
          ? DateTime.now()
          : toDateTime(docIssueDateTextController.text);
  }


  @override
  void initState() {
    DatabaseHelper helper = DatabaseHelper.instance;
    helper.fetchReg().then((success) {
      setState(() {
        _model = success;
        firstNameTextController.text = _model.firstName;
        lastNameTextController.text = _model.lastName;
        docSerialTextController.text = _model.docSerial;
        docNumberTextController.text = _model.docNum;
        docIssuedByTextController.text = _model.docIssuedBy;
        if (_model.citizenship != null)
```

```
          citizenshipTextController.text /*=
_model.citizenship.engDescription*/;
        middleNameTextController.text = _model.middleName;
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    citizenshipList.firstInit();

    return RegistrationPage(
        Form(
            key: _formKey,
            onChanged: () {
              update();
            },
            child: Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: <Widget>[
                  TitlePage(
                      'Personal data', EdgeInsets.only(bottom:
10, top: 12)),
                  PhotoPage(_model, update),
                  Padding(
                    padding: EdgeInsets.only(left: 23, bottom:
10),
                    child: FormField(
                      validator: (value) {
                        if (_model.isMale == null) return
"Select value";
                        return null;
                      },
                      builder: (FormFieldState<bool> field) {
                        return new InputDecorator(
                            decoration: InputDecoration(
                                alignLabelWithHint: true,
                                border: InputBorder.none,
                                errorText: field.errorText,
                                errorStyle: TextStyle(
                                    color: MyColor().errorColor,
                                    fontFamily: 'Roboto-
Regular',
                                    fontSize: 12)),
                            child: Container(
                              height: 25,
                              child: Row(
                                crossAxisAlignment:
CrossAxisAlignment.start,
                                children: <Widget>[
```

```
                                    CheckBox(
                                      value: _model.isMale == null
                                          ? false
                                          : _model.isMale,
                                      onChanged: () {
                                        setState(() {
                                          _model.isMale = true;
                                        });
                                      },
                                      text: 'Male',
                                      crossAxisAlignment:
                                      CrossAxisAlignment.center,
                                      iconAlignment:
Alignment.centerLeft,
                                    ),
                                    Container(
                                      padding:
EdgeInsets.only(left: 40),
                                      child: CheckBox(
                                        value: _model.isMale ==
null
                                            ? false
                                            : !_model.isMale,
                                        onChanged: () {
                                          setState(() {
                                            _model.isMale = false;
                                          });
                                        },
                                        text: 'Female',
                                        crossAxisAlignment:
                                        CrossAxisAlignment.center,
                                        iconAlignment:
Alignment.centerLeft,
                                      ),
                                    )
                                  ],
                                ),
                              ));
                        },
                      ),
                    ),
                    DecoratedTextField((String value) {
                      _model.lastName = value.trim();
                    }, 'Last name', lastNameTextController),
                    DecoratedTextField((String value) {
                      _model.firstName = value.trim();
                    }, 'Name', firstNameTextController),
                    DecoratedTextField(
                          (String value) {
                        _model.middleName = value.trim();
```

```
                    },
                    'Middle Name',
                    middleNameTextController,
                    enabled: !_model.noMiddleName,
                    padding: EdgeInsets.all(0),
                    validator: (String value) {
                      if (_model.noMiddleName) return null;
                      return Validators.latinName(value);
                    },
                  ),
                  Padding(
                    padding: _model.noMiddleName == true
                        ? EdgeInsets.symmetric(vertical: 10)
                        : EdgeInsets.only(bottom: 10),
                    child: CheckBox(
                      value: _model.noMiddleName,
                      onChanged: () {
                        setState(() {
                          _model.noMiddleName =
!_model.noMiddleName;
                          middleNameTextController.text = '';
                        });
                      },
                      text: 'I don’t middle name',
                      crossAxisAlignment:
CrossAxisAlignment.center,
                      iconAlignment: Alignment.centerLeft,
                    ),
                  ),
                  DateTextField(
                    (DateTime date) {
                      _model.birthDate = date;
                    },
                    'Birthday',
                    DateTime(1900),

maxDateBirth(docIssueDateTextController.text),
                    _model.birthDate ?? DateTime.now(),
                    birthDateTextController,
                    padding: EdgeInsets.only(bottom: 20, top:
10),
                  ),
                  Container(
                    padding: EdgeInsets.only(top: 10),
                    child: StreamBuilder(
                        stream: citizenshipList.dataStream,
                        builder: (BuildContext context,
snapShot) {
                          if (snapShot.hasData) {
                            return TypeAheadFormField(
```

```
                                autovalidate: false,
                                validator: (String value) {
                                  String error =
Validators.latinName(value);
                                  if (error == null) {
                                    for (int i = 0; i <
snapShot.data.length; i++) {
                                      if
(citizenshipTextController.text ==
snapShot.data[i].engDescription) {
                                        if
(citizenshipTextController.text ==
_model.citizenship.engDescription) {
                                          return null;
                                        }
                                        return "Select a country
name from the list";
                                      }
                                    }
                                    return "Enter a valid country
name";
                                  }
                                  return error;
                                },
                                textFieldConfiguration:
TextFieldConfiguration(
                                    autofocus: false,
                                    controller:
citizenshipTextController,
                                    maxLength: 64,
                                    decoration: InputDecoration(
                                        contentPadding:
EdgeInsets.symmetric(
                                            horizontal: 25,
vertical: 2),
                                        hintText: 'Citizenship',
                                        errorStyle: TextStyle(
                                            color:
MyColor().errorColor,
                                            fontFamily: 'Roboto-
Regular',
                                            fontSize: 12),
                                        hintStyle: TextStyle(
                                            color:
MyColor().textColor,
                                            fontFamily: 'Roboto-
Regular',
                                            fontSize: 14),
```

```
                                        focusedBorder:
OutlineInputBorder(
                                          borderRadius:
BorderRadius.circular(10),
                                          borderSide: BorderSide(
                                            color:
MyColor().buttonColor,
                                          ),
                                        ),
                                        focusedErrorBorder:
OutlineInputBorder(
                                          borderRadius:
BorderRadius.circular(10),
                                          borderSide: BorderSide(
                                            color:
MyColor().errorColor,
                                          ),
                                        ),
                                        border:
OutlineInputBorder(
                                          borderRadius:
BorderRadius.circular(10),
                                          borderSide: BorderSide(
                                            color:
MyColor().textColor,
                                          ),
                                        ),
                                        errorBorder:
OutlineInputBorder(
                                          borderRadius:
BorderRadius.circular(10),
                                          borderSide: BorderSide(
                                            color:
MyColor().errorColor,
                                          ),
                                        ))),
                                    suggestionsCallback: (pattern) {
                                      List<String> matches = List();
                                      for (int i = 0; i <
snapShot.data.length; i++)

matches.add(snapShot.data[i].engDescription);
                                      return matches.where((e) =>
e.toString().toLowerCase().contains(pattern)).toList();
                                    },
                                    itemBuilder: (context, suggestion)
{
                                      return ListTile(title:
Text(suggestion));
                                    },
```

```
                                transitionBuilder: (context,
suggestionsBox, controller) {
                                  return suggestionsBox;
                                },
                                onSuggestionSelected: (suggestion)
{
                                  citizenshipTextController.text =
suggestion;
                                  setState(() {
                                    _model.citizenship =
snapShot.data.where((citizenship) =>
citizenship.engDescription.toString().contains(suggestion)).firs
t;
                                    _model.docType =

_model.citizenship.citizenshipTypeCode ==

snapShot.data[2].citizenshipTypeCode
                                        ? docTypeRus[0]
                                        : docTypeEng[0];
                                    docSerialTextController.text =
'';

                                    docNumberTextController.text =
'';

                                  });
                                },
                              );
                            } else {
                              return Center(

child:CircularProgressIndicator()
                              );
                            }
                          }
                      ),
                    ),
                    Container(
                      padding: EdgeInsets.only(bottom:
_model.citizenship != null ? 30 : 0, top: 10),
                      child: _model.citizenship != null ?
FormField<DocType>(
                        builder: (FormFieldState<DocType> state) {
                          return InputDecorator(
                            decoration: InputDecoration(
                                contentPadding:
                                EdgeInsets.only(left: 18, top: 2,
bottom: 2),
                                hintText: 'Type of document',
                                hintStyle: TextStyle(
                                    color: MyColor().textColor,
```

```
                                    fontFamily: 'Roboto-Regular',
                                    fontSize: 14),
                                focusedBorder: OutlineInputBorder(
                                  borderRadius:
BorderRadius.circular(10),
                                  borderSide: BorderSide(
                                    color: MyColor().buttonColor,
                                  ),
                                ),
                                border: OutlineInputBorder(
                                  borderRadius:
BorderRadius.circular(10),
                                  borderSide: BorderSide(
                                    color: MyColor().textColor,
                                  ),
                                ),
                                errorBorder: OutlineInputBorder(
                                  borderRadius:
BorderRadius.circular(10),
                                  borderSide: BorderSide(
                                    color: MyColor().errorColor,
                                  ),
                                )),
                            isEmpty: _model.docType == null,
                            child: DropdownButtonHideUnderline(
                                child: Container(
                                  padding: EdgeInsets.only(right:
10),
                                  child: DropdownButton<DocType>(
                                    icon: Icon(Icons.expand_more,
color: MyColor().textColor),
                                    value: _model.docType,
                                    isDense: true,
                                    onChanged: (DocType newValue)
{
                                      setState(() {
                                        _model.docType = newValue;

docSerialTextController.text = '';

docNumberTextController.text = '';
                                        state.didChange(newValue);
                                      });
                                    },
                                    items:
(_model.citizenship.citizenshipTypeCode == 'Rus' ? docTypeRus :
docTypeEng).map((DocType value) {
                                      return
DropdownMenuItem<DocType>(
                                        value: value,
```

```
                                        child: Text(value.name),
                                      );
                                    }
                                    ).toList(),
                                  ),
                                )
                              ),
                            );
                          },
                        ): Container(),
                      ),
                      SeriesAndNumber(_model,
docSerialTextController, docNumberTextController),
                      DecoratedTextField(
                            (String value) {
                          _model.docIssuedBy = value.trim();
                        },
                        'Issued by whom',
                        docIssuedByTextController,
                        withCapital: false,
                      ),
                      DateTextField((DateTime date) {
                            _model.docIssueDate = date;
                          },
                          'When issued',

minDateIssue(birthDateTextController.text),
                          DateTime.now(),
                          _model.docIssueDate ?? DateTime.now(),
                          docIssueDateTextController),
                      Container(height: 90)
                    ]
                )
            ),
            () {
              autovalid = true;
              if (_formKey.currentState.validate()) {
                try {
                  _formKey.currentState.save();
                } catch (ignore) {}
                Route route = MaterialPageRoute(
                    builder: (context) => UpdatePage(_model)
                );
                Navigator.push(context, route);
              } else {
                ErrorSnackBar(_formKey, 'There are errors in the
fields, correct to continue');
              }
            },
            2
```

```
    );
  }
}
```

Файл «reg3_page.dart»

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:flutter_masked_text/flutter_masked_text.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/views/elements/register.dart';
import 'package:reg_model/components/model/registration.dart';
import 'package:reg_model/views/elements/information_text.dart';
import
'package:reg_model/views/elements/decorated_text_filed.dart';
import 'package:reg_model/views/elements/title_page.dart';
import 'package:reg_model/components/validator.dart';
import 'package:reg_model/views/elements/check_box.dart';
import 'package:reg_model/views/new_page.dart';
import 'package:reg_model/views/elements/error_snack_bar.dart';
import 'package:reg_model/components/bd.dart';
import 'package:reg_model/components/network/otp.dart';

const int startValueTimaer = 60;

class RegistrationPage3 extends StatefulWidget {
  RegistrationPage3({Key key,}) : super(key: key);
  @override
  _RegistrationPage3State createState() =>
_RegistrationPage3State();
}

class _RegistrationPage3State extends State<RegistrationPage3> {
  bool sentCode = false;
  final _formKey = GlobalKey<FormState>();
  final TextEditingController emailTextController = new
TextEditingController();
  final TextEditingController codeSMSTextController = new
TextEditingController();
  var controllerRus = new MaskedTextController(mask:
'+7(000)000-0000');
  var controllerNotRus = new MaskedTextController(mask:
'+000000000000000');

  RegistrationModel _model = RegistrationModel();
  String code;
  int otpRqId;
  bool activTimer = false;
  bool autovalid = false;
  bool processRequest = false;
```

```
Timer timer;
int valueTimer = startValueTimaer;

void startTimer() {
  const oneSec = const Duration(seconds: 1);
  if(activTimer){
    timer = new Timer.periodic(
        oneSec,
        (Timer timer){
          setState(() {
            if (valueTimer > 1) {
              valueTimer = valueTimer - 1;
            } else {
              valueTimer = startValueTimaer;
              activTimer = false;
              timer.cancel();
            }
          });
        }
    );
  }
}

void dispose() {
  if(timer != null)
    timer.cancel();
  super.dispose();
}

@override
void initState() {
  DatabaseHelper helper = DatabaseHelper.instance;
  helper.fetchReg().then((success) {
    setState(() {
      _model = success;
      emailTextController.text = _model.email;
      if(_model.notRusMobile){
        controllerNotRus.text = _model.mobile;
      }{
        controllerRus.text = _model.mobile;
      }
    });
  });
}

@override
Widget build(BuildContext context) {
  return RegistrationPage(
      Form(
          key: _formKey,
```

```
            onChanged: () {
              setState(() {});
              if(autovalid)
                _formKey.currentState.validate();
            },
            child: Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: <Widget>[
                  TitlePage('Contact information',
EdgeInsets.only(bottom: 20,top: 12)),
                  DecoratedTextField(
                    (String value) {
                      _model.email = value.trim();
                    }, 'E-mail', emailTextController,
                    validator: (String value){
                      while (true) {
                        if (value == value.trimRight()) break;
                        value = value.substring(0, value.length
- 1);
                      }
                      var status = Validators.email(value);
                      return status;
                    },
                  ),
                  Container(
                    //padding: EdgeInsets.only(bottom: 12),
                    child: TextFormField(
                      cursorColor: MyColor().buttonColor,
                      controller: _model.notRusMobile ?
controllerNotRus : controllerRus,
                      maxLength: _model.notRusMobile ? 16 : 15,
                      validator: (String value) {
                        return Validators.phone(value);
                      },
                      onSaved: (String value) {
                        _model.mobile =
value.trim().replaceAll('+','').replaceAll('(','').replaceAll(')
','').replaceAll('-','');
                      },
                      decoration: InputDecoration(
                          contentPadding:
EdgeInsets.symmetric(horizontal: 25, vertical: 2),
                          hintText: 'Number',
                          errorStyle: TextStyle(
                              color: MyColor().errorColor,
                              fontFamily: 'Roboto-Regular',
                              fontSize: 12),
                          hintStyle: TextStyle(
                              color: MyColor().textColor,
                              fontFamily: 'Roboto-Regular',
```

```
                                fontSize: 14),
                            focusedBorder: OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(10),
                              borderSide: BorderSide(
                                color: MyColor().buttonColor,
                              ),
                            ),
                            focusedErrorBorder:
OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(10),
                              borderSide: BorderSide(
                                color: MyColor().errorColor,
                              ),
                            ),
                            border: OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(10),
                              borderSide: BorderSide(
                                color: MyColor().textColor,
                              ),
                            ),
                            errorBorder: OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(10),
                              borderSide: BorderSide(
                                color: MyColor().errorColor,
                              ),
                            )
                        ),
                      ),
                    ),
                    CheckBox(
                      value: _model.notRusMobile,
                      onChanged: () {
                        setState(() {
                          _model.notRusMobile =
!_model.notRusMobile;
                          controllerRus = new
MaskedTextController(mask: '+7(000)000-0000');
                          controllerNotRus = new
MaskedTextController(mask: '+000000000000000');
                        });
                      },
                      text: 'Not Russian phone code',
                      crossAxisAlignment:
CrossAxisAlignment.center,
                      iconAlignment: Alignment.centerLeft,
                    ),
```

```
                    Container(
                      child: sentCode ? Column(
                        crossAxisAlignment:
CrossAxisAlignment.start,
                        children: <Widget>[
                          InformationText('Code send in your
phone'),
                          activTimer ? InformationText('You can
resend the code in ' + valueTimer.toString() + ' seconds') :
Container(),
                          Container(
                            padding: EdgeInsets.only(top: 10),
                            child: DecoratedTextField(
                                    (String value){
                                  code = value.trim();
                                },
                                'Code',
                                codeSMSTextController,
                                maxLength: 6,
                                keyboardType:
TextInputType.number,
                                validator: (String value){
                                  return;
                                },
                            )
                          )
                        ],
                      ): Container(
                          alignment: Alignment.center,
                          child : processRequest ?
CircularProgressIndicator(
                            valueColor: new
AlwaysStoppedAnimation<Color>(MyColor().buttonColor),
                          ) : null
                      ),
                    ),
                    Container(
                      height: 15,
                      alignment: Alignment.centerRight,
                      child: FlatButton(
                        child: Text(
                          'Send Code',
                          style: TextStyle(
                              color: processRequest || activTimer
? MyColor().textColor : MyColor().previousStepColor,
                              fontFamily: 'Roboto-Regular',
                              fontSize: 14
                          ),
                        ),
                        onPressed: () {
```

```
                            autovalid = true;
                            if(activTimer) return;
                            if (_formKey.currentState.validate()) {
                              try {
                                _formKey.currentState.save();
                              } catch (ignore) {}
                              setState(() {
                                processRequest = true;
                              });

Otp(_model.mobile).fetchOtp().then((success) async {
                                if(success.resultCode == 0){

Otp(_model.mobile).saveResponseData(success);
                                  otpRqId = success.otpRqId;
                                  setState(() {
                                    processRequest = false;
                                    sentCode = true;
                                    activTimer = true;
                                    startTimer();
                                    codeSMSTextController.text = '';
                                  });
                                } else{
                                  ErrorSnackBar(_formKey,
success.desc);
                                }

                              }).catchError((e) {
                                print(e.toString());
                                ErrorSnackBar(_formKey,
e.toString());
                              });
                            }
                          },
                        ),
                      ),
                      Container( height: 100)
                    ]
                  )
              ),
              () {
                if(sentCode){
                  if (_formKey.currentState.validate()) {
                    try {
                      _formKey.currentState.save();
                    } catch (ignore) {}
                    if(codeSMSTextController.text.length == 6){
                      setState(() {
                        sentCode = false;
                        valueTimer = startValueTimaer;
```

```
                  activTimer = false;
                  timer.cancel();
                  codeSMSTextController.text = '';
                });
                Route route = MaterialPageRoute(
                    builder: (context) => NewPage(_formKey,
_model, otpRqId ,code)
                );
                Navigator.push(context, route);
              } else {
                ErrorSnackBar(_formKey, 'Enter a code');
              }
            } else {
              ErrorSnackBar(_formKey, 'There are errors in the
fields, correct to continue');
            }
          }
        },
        3,
        activeButton: sentCode
    );
  }
}
```

Файл « reg4_page.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/views/finish_screen.dart';
import 'package:reg_model/views/elements/information_text.dart';
import 'package:reg_model/views/elements/caption_text.dart';
import 'package:reg_model/views/elements/title_page.dart';
import 'package:reg_model/components/model/registration.dart';
import
'package:reg_model/components/user_data_stream_builder.dart';
import 'package:reg_model/views/elements/register.dart';

class RegistrationPage4 extends StatefulWidget {
  RegistrationPage4({Key key,}) : super(key: key);
  @override
  _RegistrationPage4State createState() =>
_RegistrationPage4State();
}

class _RegistrationPage4State extends State<RegistrationPage4> {
  RegistrationModel _model = RegistrationModel();
  final UserDataStreamBuilder userData = new
UserDataStreamBuilder();

  @override
```

```
  Widget build(BuildContext context) {
    userData.firstInit();
    return RegistrationPage(
        SingleChildScrollView(
          child: Scrollbar(
            child:  Container(
              child:StreamBuilder(
                  stream: userData.dataStream,
                  builder: (BuildContext context, snapShot) {
                    if (snapShot.hasData) {
                      _model=snapShot.data;
                      return Column(
                        crossAxisAlignment:
CrossAxisAlignment.stretch,
                        children: <Widget>[
                          TitlePage('Checking the entered data',
EdgeInsets.only(bottom: 20,top: 12)),
                          CaptionText('Name'),
                          InformationText(_model.lastName + ' '
+ _model.firstName + ' ' + _model.middleName ?? ''),
                          CaptionText('Birthday'),

InformationText(toString(_model.birthDate)),
                          CaptionText('Citizenship'),

InformationText(_model.citizenship.engDescription),
                          CaptionText('Type of document'),
                          InformationText(_model.docType.name),
                          CaptionText('Series and Number of
document'),
                          InformationText(_model.docSerial !=
null ? _model.docSerial+ ' ' + _model.docNum : _model.docNum) ,
                          CaptionText('When issued'),
                          InformationText(_model.docIssuedBy),
                          CaptionText('Issued by whom'),

InformationText(toString(_model.docIssueDate)),
                          CaptionText('E-mail'),
                          InformationText(_model.email),
                          CaptionText('Number'),
                          InformationText(_model.mobile),
                          Container( height: 100)
                        ],
                      );
                    }else {
                      return Container(
                        child: CircularProgressIndicator(
                          //backgroundColor:
MyColor().buttonColor,
```

```
                            valueColor: new
AlwaysStoppedAnimation<Color>(MyColor().buttonColor),
                          ),
                        );
                      }
                    }
                ),
              ),
            )
          ), (){
            Route route =MaterialPageRoute(
                builder: (context) => FinishPage()
            );
            Navigator.push(context, route);
          },
          4
      );
    }
}
```

Файл «update_page.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/views/elements/proces_request.dart';
import 'package:reg_model/components/model/registration.dart';
import 'package:reg_model/components/bd.dart';
import 'package:reg_model/views/reg3_page.dart';

class UpdatePage extends StatefulWidget {
  RegistrationModel model;
  UpdatePage(this.model);

  @override
  _UpdatePageState createState() => _UpdatePageState();
}

class _UpdatePageState extends State<UpdatePage> {
  @override
  void initState() {
    super.initState();
    DatabaseHelper helper = DatabaseHelper.instance;
    helper.upsertReg(widget.model).then((model){
      Navigator.pop(context);
      Route route = MaterialPageRoute(
        builder: (context) => RegistrationPage3(),
      );
      Navigator.push(context, route);
    });
  }
```

```
  @override
  Widget build(BuildContext context) {
    return ProcesRequest();
  }
}
```

Файл «add_photo_button.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/views/elements/menu_add_photo.dart';

class AddPhotoButton extends StatelessWidget{
  bool withCloseWindow;
  var getImage;

  AddPhotoButton(this.getImage, {this. withCloseWindow :
false});

  @override
  Widget build(BuildContext context) {
    return Container(
      height: 30,
      child: FlatButton(
        child: Text(
          'Add Photo',
          style: TextStyle(
              color: MyColor().previousStepColor,
              fontFamily: 'Roboto-Regular',
              fontSize: 14
          ),
        ),
        onPressed: (){
          if(withCloseWindow)
            Navigator.of(context).pop();

          return showDialog(
              context: context,
              builder: (BuildContext context) {
                return MenuAddPhoto(getImage);
              }
          );
        },
      ),
    );
  }
}
```

Файл «cancel_button.dart »

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';

class CancelButton extends StatelessWidget{

  @override
  Widget build(BuildContext context) {
    return FlatButton(
      child: Text(
        'Cancel',
        style: TextStyle(
            color: MyColor().textColor,
            fontFamily:
            'Roboto-Regular',
            fontSize: 14),
      ),
      onPressed: () {
        Navigator.of(context).pop();
      },
    );
  }
}
```

Файл «caption_text.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';

class CaptionText extends StatelessWidget{
  String text;
  CaptionText(this.text);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: EdgeInsets.symmetric(vertical: 5),
      child: Text(
        text,
        style: TextStyle(
            color: MyColor().captionTextColor,
            fontFamily: 'Roboto-Regular',
            fontSize: 14
        ),
      ),
    );
  }
}
```

Файл «check_box.dart»

```
import 'package:flutter/material.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:reg_model/step.dart';
import 'package:url_launcher/url_launcher.dart';

class CheckBox extends StatelessWidget{
  bool value;
  var onChanged;
  String text;
  CrossAxisAlignment crossAxisAlignment;
  Alignment iconAlignment;

  CheckBox({this.value, this.onChanged, this.text,
this.crossAxisAlignment, this.iconAlignment});

  @override
  Widget build(BuildContext context) {
    var textSpan = new List<TextSpan>();
    List <String> partsText = text.split('[');
    for(int i = 0; i < partsText.length; i++){
      if(partsText[i].contains(']]')){
        textSpan[textSpan.length-1] = new TextSpan(
            text: partsText[i-1],
            style: TextStyle(
                color: MyColor().previousStepColor,
                fontFamily: 'Roboto-Regular',
                fontSize: 14
            ),
            recognizer: new TapGestureRecognizer()
              ..onTap = (() async {
                if (await canLaunch(partsText[i])) {
                  await launch(partsText[i]);
                } else {
                  throw 'Could not launch $partsText[i]';
                }
              }
            )
        );
        List <String> link = partsText[i].split(']]');
        partsText.add('');
        for(int j = partsText.length-1; j > i; j--){
          partsText[j] = partsText[j-1];
        }
        partsText[i] = link[0];
        partsText[i+1] = link[1];
      }else{
        textSpan.add(
          new TextSpan(
            text: partsText[i],
```

```
              style: TextStyle(
                  color: MyColor().textColor,
                  fontFamily: 'Roboto-Regular',
                  fontSize: 14
              )
            )
          );
        }
      }

      return Row(
        crossAxisAlignment: crossAxisAlignment,
        children: <Widget>[
          IconButton(
              padding: EdgeInsets.only(left: 0),
              alignment: iconAlignment,
              icon: SvgPicture.asset(
                  'assets/icons/checkBox.svg',
                  width: 25,
                  height: 25,
                  color: value == true? MyColor().activeStepColor
: MyColor().notActiveStepColor
              ),
              onPressed: (){
                onChanged();
              }
          ),
          RichText(
            text: new TextSpan(
              children: textSpan,
            ),
          )
        ],
      );
    }
}
```

Файл «date_text_field.dart»

```
import 'dart:io' show Platform;
import 'package:intl/intl.dart';
import 'package:flutter/material.dart';
import 'package:flutter_masked_text/flutter_masked_text.dart';
import
'package:flutter_rounded_date_picker/src/material_rounded_date_p
icker_style.dart';
import
'package:flutter_rounded_date_picker/src/material_rounded_year_p
icker_style.dart';
import
```

```
'package:reg_model/flutter_rounded_date_picker/flutter_rounded_d
ate_picker_widget.dart';
import
'package:reg_model/flutter_cupertino_date_picker/date_picker.dar
t';
import 'package:reg_model/components/validator.dart';
import 'package:reg_model/step.dart';

class DateTextField extends StatefulWidget {
  DateTime minDate = DateTime(1900),
      maxDate = DateTime.now(),
      initialValue = DateTime.now();
  String hintText;
  MaskedTextController textController;
  var onSaved;
  EdgeInsetsGeometry padding;

  DateTextField(this.onSaved, this.hintText, this.minDate,
this.maxDate,
      this.initialValue, this.textController,
      {this.padding: const EdgeInsets.only(bottom: 20), Key
key})
      : super(key: key);

  @override
  _DateTextFieldState createState() => _DateTextFieldState();
}

class _DateTextFieldState extends State<DateTextField> {
  bool activeButton = false;

  @override
  Widget build(BuildContext context) {
    return Container(
        //padding: widget.padding,
        child: Stack(
          alignment: Alignment.topCenter,
          children: <Widget>[
            TextFormField(
              cursorColor: MyColor().buttonColor,
              controller: widget.textController,
              maxLength: 10,
              validator: (String value) {
                String validatorDate =
                    Validators.isDate(value, widget.minDate,
widget.maxDate);
                if (validatorDate == null) {
                  widget.initialValue = toDateTime(value);
                }
                return validatorDate;
```

```
                },
                onSaved: (String value) {
                  widget.initialValue = toDateTime(value);
                  widget.onSaved(widget.initialValue);
                },
                decoration: InputDecoration(
                    contentPadding:
                        EdgeInsets.symmetric(horizontal: 25,
vertical: 2),
                    hintText: widget.hintText,
                    hintStyle: TextStyle(
                        color: MyColor().textColor,
                        fontFamily: 'Roboto-Regular',
                        fontSize: 14),
                    errorStyle: TextStyle(
                        color: MyColor().errorColor,
                        fontFamily: 'Roboto-Regular',
                        fontSize: 12),
                    focusedBorder: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(10),
                      borderSide: BorderSide(
                        color: MyColor().buttonColor,
                      ),
                    ),
                    focusedErrorBorder: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(10),
                      borderSide: BorderSide(
                        color: MyColor().errorColor,
                      ),
                    ),
                    border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(10),
                      borderSide: BorderSide(
                        color: MyColor().textColor,
                      ),
                    ),
                    errorBorder: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(10),
                      borderSide: BorderSide(
                        color: MyColor().errorColor,
                      ),
                    )),
              ),
              Container(
                  alignment: Alignment.centerRight,
                  padding: EdgeInsets.only(left: 30),
                  child: IconButton(
                    icon: Icon(
                      Icons.date_range,
                      color: activeButton
```

```
                        ? MyColor().notActiveStepColor
                        : MyColor().textColor,
                  ),
                  onPressed: () async {
                    setState(() {
                      activeButton = true;
                    });
                    String validatorDate = Validators.isDate(
                        widget.textController.text,
                        widget.minDate,
                        widget.maxDate);
                    if (validatorDate == null) {
                      widget.initialValue =
toDateTime(widget.textController.text);
                    }
                    if (Platform.isAndroid) {
                      DateTime picked = await
showRoundedDatePicker(
                          context: context,
                          initialDate: widget.initialValue,
                          firstDate: DateTime(1900),
                          lastDate: DateTime.now(),
                          fontFamily: 'Roboto-Regular',
                          styleYearPicker:
MaterialRoundedYearPickerStyle(
                              textStyleYear:
                                  TextStyle(color:
MyColor().textColor)),
                          styleDatePicker:
MaterialRoundedDatePickerStyle(
                            paddingDateYearHeader:
EdgeInsets.symmetric(
                                vertical: 20, horizontal: 25),
                            paddingMonthHeader:
EdgeInsets.symmetric(
                                vertical: 15, horizontal: 25),
                            paddingDatePicker:
EdgeInsets.symmetric(
                                vertical: 0, horizontal: 17),
                            textStyleYearButton: TextStyle(
                              color:
MyColor().activeButtonColor,
                              fontFamily: 'Roboto-Regular',
                            ),
                            textStyleDayButton: TextStyle(
                                color:
MyColor().numberStepColor,
                                fontFamily: 'Roboto-Regular',
                                fontSize: 24),
                            colorArrowPrevious:
```

```
MyColor().headerColor,
                              colorArrowNext:
MyColor().headerColor,
                              textStyleMonthYearHeader: TextStyle(
                                color: MyColor().buttonColor,
                                fontFamily: 'Roboto-Regular',
                                fontSize: 14,
                              ),
                              textStyleDayHeader: TextStyle(
                                  color: MyColor().textColorTwo,
                                  fontFamily: 'Roboto-Regular',
                                  fontSize: 14),
                              backgroundPicker:
MyColor().backgroundColor,
                              textStyleDayOnCalendar: TextStyle(
                                  fontFamily: 'Roboto-Regular',
                                  color: MyColor().textColor),
                              textStyleDayOnCalendarDisabled:
TextStyle(
                                  fontFamily: 'Roboto-Regular',
                                  color: MyColor().textColorTwo),
                              textStyleCurrentDayOnCalendar:
TextStyle(
                                fontFamily: 'Roboto-Regular',
                                color: MyColor().buttonColor,
                              ),
                              decorationDateSelected:
BoxDecoration(
                                  color: MyColor().headerColor,
                                  shape: BoxShape.circle),
                              textStyleDayOnCalendarSelected:
TextStyle(
                                  fontFamily: 'Roboto-Regular',
                                  color:
MyColor().backgroundColor),
                              textStyleButtonPositive: TextStyle(
                                  fontFamily: 'Roboto-Regular',
                                  color: MyColor().buttonColor),
                              textStyleButtonNegative: TextStyle(
                                  fontFamily: 'Roboto-Regular',
                                  color: MyColor().textColor),
                            ),
                            theme: ThemeData(
                              primaryColor: MyColor().headerColor,
                              accentColor: MyColor().buttonColor,
                            ));
                      setState(() {
                        activeButton = false;
                      });
                      if (picked != null) {
```

```
                        widget.initialValue = picked;
                        widget.textController.text =
toString(picked);
                        widget.onSaved(picked);
                      }
                    } else if (Platform.isIOS) {
                      DatePicker.showDatePicker(context,
                          minDateTime: widget.minDate,
                          maxDateTime: widget.maxDate,
                          initialDateTime: widget.initialValue,
                          dateFormat: 'd,MMMM,yyyy',
                          onChange: (dateTime, List<int> index)
{
                        setState(() {
                          widget.initialValue = dateTime;
                          widget.textController.text =
toString(DateTime(dateTime.year, dateTime.month, dateTime.day));
                          widget.onSaved(dateTime);
                        });
                      });
                    }
                    setState(() {
                      activeButton = false;
                    });
                  },
                ))
          ],
        ));
  }
}
```

Файл «decorated_text_filed.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/components/validator.dart';
import 'package:reg_model/step.dart';

class DecoratedTextField extends StatefulWidget {
  String hintText;
  TextEditingController textController;
  var onSaved;
  EdgeInsetsGeometry padding;
  int maxLength;
  bool enabled;
  TextInputType keyboardType;
  var validator;
  bool withCapital;

  DecoratedTextField(this.onSaved, this.hintText,
this.textController,
```

```
      {
        this.maxLength: 64,
        this.enabled: true,
        this.keyboardType: TextInputType.text,
        this.validator,
        this.padding: const EdgeInsets.only(bottom: 10),
        this.withCapital : true,
        Key key})
      : super(key: key);

  @override
  _DecoratedTextFieldState createState() =>
_DecoratedTextFieldState();
}

class _DecoratedTextFieldState extends State<DecoratedTextField>
{
  bool activeButton = false;

  @override
  Widget build(BuildContext context) {
    return Container(
        padding: widget.padding,
        child: TextFormField(
          cursorColor: MyColor().buttonColor,
          enabled: widget.enabled,
          controller: widget.textController,
          maxLength: widget.enabled? widget.maxLength: null,
          keyboardType: widget.keyboardType,
          textCapitalization: widget.withCapital == true ?
TextCapitalization.sentences : TextCapitalization.none,
          validator: (String value) {
            return widget.validator == null ?
Validators.latinName(value):widget.validator(value);
          },
          onSaved: (String value) {
            widget.onSaved(value);
          },
          decoration: InputDecoration(
              contentPadding: EdgeInsets.symmetric(horizontal:
25, vertical: 2),
              hintText: widget.hintText,
              hintStyle: TextStyle(
                  color: MyColor().textColor,
                  fontFamily: 'Roboto-Regular',
                  fontSize: 14),
              errorStyle: TextStyle(
                  color: MyColor().errorColor,
                  fontFamily: 'Roboto-Regular',
                  fontSize: 12),
```

```
                focusedBorder: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                  borderSide: BorderSide(
                    color: MyColor().buttonColor,
                  ),
                ),
                focusedErrorBorder: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                  borderSide: BorderSide(
                    color: MyColor().errorColor,
                  ),
                ),
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                  borderSide: BorderSide(
                    color: MyColor().textColor,
                  ),
                ),
                errorBorder: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                  borderSide: BorderSide(
                    color: MyColor().errorColor,
                  ),
                )),
        ));
  }
}
```

Файл «error_snack_bar.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';

class ErrorSnackBar{
  GlobalKey _formKey;
  String text;

  ErrorSnackBar(this._formKey, this.text){
    Scaffold.of(_formKey.currentContext).showSnackBar(SnackBar(
        backgroundColor: MyColor().errorColor,
        content: new Text(text),
        duration: new Duration(seconds: 4)));
  }
}
```

Файл «information_text.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';
```

```
class InformationText extends StatelessWidget{
  String text;
  InformationText(this.text);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: EdgeInsets.symmetric(vertical: 5),
      child: Text(
        text,
        style: TextStyle(
            color: MyColor().textColor,
            fontFamily: 'Roboto-Regular',
            fontSize: 14
        ),
      ),
    );
  }
}
```

Файл «menu_add_photo.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/views/elements/cancel_button.dart';
import 'package:image_picker/image_picker.dart';

class MenuAddPhoto extends StatelessWidget{
  double height = 30;
  var getImage;

  MenuAddPhoto(this.getImage);

  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.center,
      mainAxisAlignment: MainAxisAlignment.end,
      children: <Widget>[
        AlertDialog(
            contentPadding: EdgeInsets.symmetric(vertical: 15),
            shape: RoundedRectangleBorder(
                borderRadius: new BorderRadius.circular(10)
            ),
            content: Container(
              child: Column(
                children: <Widget>[
                  Container(
                    height: height,
                    child: FlatButton(
```

```
                        child: Text(
                          'Choose Photo',
                          style: TextStyle(
                              color: MyColor().previousStepColor,
                              fontFamily: 'Roboto-Regular',
                              fontSize: 14
                          ),
                        ),
                        onPressed: (){
                          Navigator.pop(context);
                          getImage(ImageSource.gallery);
                        },
                      ),
                    ),
                    Container(
                      height: height,
                      child: FlatButton(
                        child: Text(
                          'Create Photo',
                          style: TextStyle(
                              color: MyColor().previousStepColor,
                              fontFamily: 'Roboto-Regular',
                              fontSize: 14
                          ),
                        ),
                        onPressed: (){
                          Navigator.pop(context);
                          getImage(ImageSource.camera);
                        },
                      ),
                    ),
                    Container(
                      height: height,
                      child: CancelButton(),
                    ),
                  ],
                ),
              )
            )
          ]
      );
    }
}
```

Файл «photo_error_requirements.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/views/elements/add_photo_button.dart';
import 'package:reg_model/views/elements/cancel_button.dart';
```

```
import 'package:reg_model/components/photo.dart';

class PhotoErrorRequirementsPage extends StatelessWidget {
  var getImage;
  PhotoValidationResult errors;

  PhotoErrorRequirementsPage(this.getImage, this.errors);

  @override
  Widget build(BuildContext context) {
    List<Widget> errorsList = new List<Widget>();

    for (var error in errors.failures) {
      errorsList.add(Container(
          padding: EdgeInsets.only(bottom: 20),
          alignment: Alignment.centerLeft,
          child: Text(
            error.desc,
            style: TextStyle(
                color: MyColor().errorColor,
                fontFamily: 'Roboto-Regular',
                fontSize: 14),
          )));
    }

    return AlertDialog(
      shape:
          RoundedRectangleBorder(borderRadius: new
BorderRadius.circular(15)),
      title: Center(
        child: Text(
          'Photo requirements',
          style: TextStyle(
              color: MyColor().headerColor,
              fontFamily: 'Roboto-Black',
              fontSize: 14),
        ),
      ),
      content: Container(
          height: 300,
          padding: EdgeInsets.symmetric(horizontal: 20),
          child: Column(
            children: <Widget>[
              Container(
                alignment: Alignment.centerLeft,
                padding: EdgeInsets.only(bottom: 20),
                child: Text(
                  'The photo doesn’t meet\nrequirements',
                  style: TextStyle(
                      color: MyColor().textColor,
```

```
                        fontFamily: 'Roboto-Regular',
                        fontSize: 14),
                  ),
              ),
              Expanded(
                  child: SingleChildScrollView(
                      child: Column(children: errorsList))),
              Container(
                  padding: EdgeInsets.only(top: 10),
                  child: Row(
                    mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                    children: <Widget>[
                      Container(
                        padding: EdgeInsets.only(left: 1),
                        alignment: Alignment.centerLeft,
                        child: CancelButton(),
                      ),
                      AddPhotoButton(getImage, withCloseWindow:
true)
                    ],
                  ))
            ],
          )),
    );
  }
}
```

Файл «photo_page.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/views/elements/add_photo_button.dart';
import 'package:reg_model/views/elements/menu_add_photo.dart';
import
'package:reg_model/views/elements/photo_requirements_page.dart';
import
'package:reg_model/views/elements/photo_error_requirements_page.
dart';
import 'package:reg_model/components/model/registration.dart';
import 'package:reg_model/components/photo.dart';
import 'dart:async';
import 'dart:convert';
import 'dart:core';
import 'dart:io';
import 'package:flutter/rendering.dart';
import 'package:image/image.dart' as img;
import 'package:image_cropper/image_cropper.dart';
import 'package:image_picker/image_picker.dart';
```

```
final String serviceUrl = "https://ph.dev.iteco.mobi/";

class PhotoPage extends StatefulWidget {
  RegistrationModel model;
  var update;

  PhotoPage(
    this.model,
    this.update, {
    Key key,
  }) : super(key: key);

  @override
  _PhotoPageState createState() => _PhotoPageState();
}

class _PhotoPageState extends State<PhotoPage> {
  final _formKey = GlobalKey<FormState>();
  MemoryImage loadedImage;
  MemoryImage previousLoadedImage;
  bool photoUpload = false;

  Future getImage(ImageSource source) async {
    File originalImage = await ImagePicker.pickImage(source:
source, maxWidth: 800, maxHeight: 800);
    if(originalImage != null){
      if(originalImage.path != null){
        File image = await ImageCropper.cropImage(
            sourcePath: originalImage.path,
            maxWidth: 800,
            maxHeight: 800,
            cropStyle: CropStyle.rectangle,
            androidUiSettings: AndroidUiSettings(
              lockAspectRatio: false,
              toolbarTitle: 'Edit photo',
            ),
            iosUiSettings: IOSUiSettings(
                showActivitySheetOnDone: false,
                aspectRatioLockDimensionSwapEnabled: true,
                aspectRatioLockEnabled: false,
                hidesNavigationBar: false,
                doneButtonTitle: 'OK',
                cancelButtonTitle: 'Cancel'));
        bool ChangePhoto = false;

        if (image == null) return null;
        if (loadedImage != null) {
          ChangePhoto = true;
          previousLoadedImage = loadedImage;
          loadedImage = null;
```

```
          widget.update();
        }

        photoUpload = true;
        widget.update();

        List<int> imageBytes = image.readAsBytesSync();

        PhotoCheckerV2 photoCheckerV2 = new PhotoCheckerV2();
        photoCheckerV2.setServiceURL(serviceUrl);
        photoCheckerV2
            .validatePhoto(imageBytes)
            .whenComplete(() =>
Scaffold.of(context).hideCurrentSnackBar())
            .then((PhotoValidationResult v) {
          if (v.failures != null && v.failures.length > 0) {
            if (ChangePhoto == true) {
              loadedImage = previousLoadedImage;
              previousLoadedImage = null;
              widget.update();
            }
            String photo = base64.encode(imageBytes);
            print(photo);
            photoUpload = false;
            widget.update();

            showDialog(
                context: context,
                builder: (_) =>
PhotoErrorRequirementsPage(getImage, v));
            return;
          }
          img.Image src = img.decodeImage(imageBytes);
          img.Image crop = img.copyCrop(src, v.x, v.y, v.width,
v.height);
          imageBytes = img.encodeJpg(crop);

          photoUpload = false;
          previousLoadedImage = null;
          loadedImage = MemoryImage(imageBytes);
          widget.update();

          setState(() {
            widget.model.photo = imageBytes;
          });
          String photo = base64.encode(imageBytes);
          print(photo);
        }).catchError((e, stackTrace) {
          photoUpload = false;
          widget.update();
```

```
          print(stackTrace);

Scaffold.of(_formKey.currentContext).showSnackBar(SnackBar(
              backgroundColor: MyColor().errorColor,
              content: new Text("Service error, please try again
later"),
              duration: new Duration(seconds: 4)));
        });
        return;
      }
    }
    Scaffold.of(_formKey.currentContext).showSnackBar(SnackBar(
        backgroundColor: MyColor().errorColor,
        content: new Text('No photo selected'),
        duration: new Duration(seconds: 4)));
    return null;
  }

  @override
  Widget build(BuildContext context) {
    return FormField(
      validator: (value) {
        if (loadedImage == null) return "Select value";
        return null;
      },
      builder: (FormFieldState<bool> field) {
        return new InputDecorator(
            decoration: InputDecoration(
                alignLabelWithHint: true,
                border: InputBorder.none,
                errorText: field.errorText,
                errorStyle: TextStyle(
                    color: MyColor().errorColor,
                    fontFamily: 'Roboto-Regular',
                    fontSize: 12)),
            child: Container(
              padding: EdgeInsets.all(0),
              child: Form(
                  key: _formKey,
                  child: Row(
                    children: <Widget>[
                      Container(
                          child: loadedImage == null
                              ?
                          ButtonTheme(
                            minWidth: 100,
                            height: 125,
                            shape: RoundedRectangleBorder(
                                borderRadius:
BorderRadius.circular(10)),
```

```
                              child: RaisedButton(
                                elevation: 0,
                                color:
MyColor().activeButtonColor,
                                child: !photoUpload ? Icon(
                                  Icons.photo_camera,
                                  color: MyColor().buttonColor,
                                  size: 34,
                                ) : Container(
                                  child:
CircularProgressIndicator(
                                    //backgroundColor:
MyColor().buttonColor,
                                    valueColor: new
AlwaysStoppedAnimation<Color>(MyColor().buttonColor),
                                  ),
                                ),
                                onPressed: () {
                                  return showDialog(
                                      context: context,
                                      builder: (BuildContext
context) {
                                        return
MenuAddPhoto(getImage);
                                      });
                                },
                              ),
                            ) : Container(
                                width: 100,
                                height: 125,
                                alignment: Alignment.center,
                                decoration: new BoxDecoration(
                                    borderRadius:
BorderRadius.circular(10),
                                    image: DecorationImage(
                                        image: loadedImage, fit:
BoxFit.fill)))
                        ),
                        Container(
                          padding: EdgeInsets.only(left: 10),
                          child: Column(
                            crossAxisAlignment:
CrossAxisAlignment.start,
                            children: <Widget>[
                              AddPhotoButton(getImage),
                              Container(
                                height: 25,
                                child: FlatButton(
                                  child: Text(
                                    'Photo requirements',
```

```
                                    style: TextStyle(
                                        color:
MyColor().textColor,
                                        fontFamily: 'Roboto-
Regular',
                                        fontSize: 13),
                                  ),
                                  onPressed: () {
                                    return showDialog<void>(
                                      context: context,
                                      builder: (BuildContext
context) {
                                        return
PhotoRequirementsPage(getImage);
                                      },
                                    );
                                  },
                                ),
                              )
                            ],
                          ),
                        )
                      ],
                    )),
              )
          );
        },
    );
  }
}
```

Файл «photo_requirements_page.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/views/elements/add_photo_button.dart';
import 'package:reg_model/views/elements/cancel_button.dart';

class PhotoRequirementsPage extends StatelessWidget{
  var getImage;

  PhotoRequirementsPage(this.getImage);

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      shape: RoundedRectangleBorder(
          borderRadius: new BorderRadius.circular(15)),
      title: Center(
        child: Text(
```

```
        'Photo requirements',
        style: TextStyle(
            color: MyColor().headerColor,
            fontFamily: 'Roboto-Black',
            fontSize: 14),
      ),
    ),
    content: Container(
        padding: EdgeInsets.symmetric(horizontal: 20),
        child: Column(
          mainAxisAlignment:
          MainAxisAlignment.spaceBetween,
          children: <Widget>[
            Container(
                padding:
                EdgeInsets.symmetric(vertical: 10),
                width: 100,
                height: 120,
                decoration: BoxDecoration(
                    image: DecorationImage(
                      image: AssetImage(
                          'assets/images/sample_photo.jpg'),
                      fit: BoxFit.fitHeight,
                    ),
                    borderRadius: BorderRadius.all(
                        Radius.circular(15)
                    )
                )
            ),
            Container(
              alignment: Alignment.center,
              padding: EdgeInsets.only(bottom: 20),
              child: Text(
                'Example',
                style: TextStyle(
                    color: MyColor().headerColor,
                    fontFamily: 'Roboto-Regular',
                    fontSize: 14),
              ),
            ),
            Expanded(
                flex: 1,
                child: new SingleChildScrollView(
                    child: Scrollbar(
                      child: Column(
                        children: <Widget>[
                        Text(
                            'The photo must',
                            style: TextStyle(
                                color: MyColor().textColor,
```

```
                                    fontFamily: 'Roboto-Regular',
                                    fontSize: 14),
                                textAlign: TextAlign.justify)
                            ]
                          ),
                        )
                    )
                ),
                Container(
                    padding: EdgeInsets.only(top: 10),
                    child: Row(
                      mainAxisAlignment:
                      MainAxisAlignment.spaceBetween,
                      children: <Widget>[
                        Container(
                          padding: EdgeInsets.only(left: 1),
                          alignment: Alignment.centerLeft,
                          child: CancelButton(),
                        ),
                        AddPhotoButton(getImage, withCloseWindow:
true)
                      ],
                    )
                )
              ],
            )
        ),
    );
  }
}
```

Файл «process_request.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';

class ProcesRequest extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
        decoration: BoxDecoration(
          color: MyColor().backgroundColor,
        ),
        child: Center(
          child: CircularProgressIndicator(
               valueColor: new
AlwaysStoppedAnimation<Color>(MyColor().buttonColor),
          )
        )
    );
```

```
  }
}
```

Файл «register.dart»

```
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/views/elements/step_informer.dart';

class RegistrationPage extends StatefulWidget {
  Widget body;
  var onPressed;
  int activeStep;
  bool activeButton;
  RegistrationPage(this.body, this.onPressed, this.activeStep,
{this.activeButton: true, Key key,}) : super(key: key);
  @override
  _RegistrationPageState createState() =>
_RegistrationPageState();
}

class _RegistrationPageState extends State<RegistrationPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        elevation: 0,
        backgroundColor: MyColor().backgroundColor,
        centerTitle: true,
        title: Text(
          'Registration',
          style: TextStyle(
              color: MyColor().textColor,
              fontFamily: 'Roboto-Black',
              fontSize: 14
          ),
        ),
        leading: IconButton(
          icon: SvgPicture.asset('assets/icons/arrow_left.svg'),
          onPressed: (){
            Navigator.pop(context);
          },
        ),
      ),
      body: Stack(
        fit: StackFit.expand,
        children: <Widget>[
          DecoratedBox(
            position: DecorationPosition.background,
```

```
            decoration: BoxDecoration(
                image: DecorationImage(
                  image:
AssetImage('assets/images/reg_background.png'),
                  fit: BoxFit.fill,
                )
            ),
          ),
          Container(
            child: SingleChildScrollView(
              child: Scrollbar(
                child:  Container(
                  child: Column(
                    crossAxisAlignment:
CrossAxisAlignment.start,
                    children: <Widget>[
                      Center(
                        child: StepInformer(
                            stepsCount: 4,
                            activeStep: widget.activeStep
                        ),
                      ),
                      Container(
                        padding:
EdgeInsets.symmetric(horizontal: 45),
                        child: Container(
                          child: widget.body,
                        )/*Column(
                          crossAxisAlignment:
CrossAxisAlignment.stretch,
                          children: widget.body,
                        )*/,
                      )
                    ],
                  ),
                ),
              ),
            ),
          )
        ],
      ),
      floatingActionButton: Container(
        child:  FloatingActionButton(
          elevation: 0,
          onPressed: widget.onPressed,
          backgroundColor: widget.activeButton == true?
MyColor().activeButtonColor: MyColor().notActiveButtonColor,
          child: Center(
            child:
SvgPicture.asset('assets/icons/arrow_right.svg',
```

```
                color: widget.activeButton == true?
MyColor().buttonColor: MyColor().numberStepColor,
              ),
            ),
          ),
        ),
      );
    }
}
```

Файл «series_number.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/components/model/registration.dart';
import
'package:reg_model/views/elements/decorated_text_filed.dart';
import 'package:reg_model/components/validator.dart';

class SeriesAndNumber extends StatelessWidget{
  RegistrationModel _model;
  TextEditingController docSerialTextController;
  TextEditingController docNumberTextController;

  SeriesAndNumber(this._model, this.docSerialTextController,
this.docNumberTextController);

  @override
  Widget build(BuildContext context) {
    if(_model.docType != null){
      switch (_model.docType.id) {
        case 21:
          return Row(
              children: <Widget>[
                Expanded(
                  child: Container(
                    padding: EdgeInsets.only(right: 10),
                    child: DecoratedTextField(
                      (String value){
                        _model.docSerial = value.trim();
                      },
                      'Series',
                      docSerialTextController,
                      maxLength: 4,
                      keyboardType: TextInputType.number,
                      validator: (value) {
                        return Validators.serial(value,4);
                      },
                    ),
                  ),
                ),
```

```
            Expanded(
                child: DecoratedTextField(
                    (String value){
                      _model.docNum = value.trim();
                    },
                    'Number',
                    docNumberTextController,
                    maxLength: 6,
                    keyboardType: TextInputType.number,
                    validator: (value) {
                      return Validators.number(value, 6);
                    }
                )
            )
          ]
      );
    case 22:
      return Row(
          children: <Widget>[
            Expanded(
              child: Container(
                padding: EdgeInsets.only(right: 10),
                child: DecoratedTextField(
                  (String value){
                    _model.docSerial = value.trim();
                  },
                  'Series',
                  docSerialTextController,
                  maxLength: 2,
                  keyboardType: TextInputType.number,
                  validator: (value) {
                    return Validators.serial(value,2);
                  },
                ),
              ),
            ),
            Expanded(
                child: DecoratedTextField(
                    (String value){
                      _model.docNum = value.trim();
                    },
                    'Number',
                    docNumberTextController,
                    maxLength: 7,
                    keyboardType: TextInputType.number,
                    validator: (value) {
                      return Validators.number(value, 7);
                    }
                )
            )
```

```
          ]
      );
    case 3:
      return Row(
          children: <Widget>[
            Expanded(
              child: Container(
                padding: EdgeInsets.only(right: 10),
                child: DecoratedTextField(
                  (String value){
                    _model.docSerial = value.trim();
                  },
                  'Series',
                  docSerialTextController,
                  maxLength: 9,
                  validator: (value) {
                    return Validators.seralSR(value);
                  },
                ),
              ),
            ),
            Expanded(
                child: DecoratedTextField(
                    (String value){
                      _model.docNum = value.trim();
                    },
                    'Number',
                    docNumberTextController,
                    maxLength: 6,
                    keyboardType: TextInputType.number,
                    validator: (value) {
                      return Validators.number(value, 6);
                    }
                )
            )
          ]
      );
    case 11:
      return DecoratedTextField(
        (String value) {
          _model.docSerial = null;
          _model.docNum = value.trim();
        },
        'Number',
        docNumberTextController,
        keyboardType: TextInputType.text,
        validator: Validators.notEmpty,
      );

    default:
```

```
            break;
        }
      }
      return Container();
    }
}
```

Файл «step_informer.dart»

```
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:reg_model/step.dart';

class StepInformer extends StatelessWidget {
  StepInformer({Key key, this.stepsCount, this.activeStep}) :
super(key: key);

  final int stepsCount;
  int activeStep;

  @override
  Widget build(BuildContext context) {
    List<Widget> childs = new List<Widget>();
    for (int n = 1; n <= stepsCount; ++n) {
      childs.add(
          new Container(
            child: Stack(
                alignment: Alignment.center,
                children: <Widget>[
                  SvgPicture.asset(
                      'assets/icons/step.svg',
                      width: 40,
                      height: 40,
                      color: n < activeStep ?
MyColor().previousStepColor : n == activeStep ?
MyColor().activeStepColor : MyColor().notActiveStepColor
                  ),
                  Text(
                      n.toString(),
                      style: TextStyle(
                          color: MyColor().numberStepColor,
                          fontFamily: 'Roboto-Black',
                          fontSize: 14
                      )
                  ),
                ]
            ),
          )
      );
      if (n != stepsCount) {
```

```
        childs.add(new Padding(padding:
EdgeInsets.symmetric(horizontal: 4),
          child: Container(
            decoration: BoxDecoration(
                color: n < activeStep ?
MyColor().previousStepColor : MyColor().notActiveStepColor,
                borderRadius: BorderRadius.circular(5)
            ),
            width: 12,
            height: 3,
          ),
        ));
      }
    }
    return ConstrainedBox(
        constraints: const BoxConstraints(
            minWidth: 300,
            minHeight: 100
        ),
        child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: childs
        )
    );
  }
}
```

Файл «title_page.dart»

```
import 'package:flutter/material.dart';
import 'package:reg_model/step.dart';

class TitlePage extends StatelessWidget{
  String text;
  EdgeInsets padding;
  TitlePage(this.text,this.padding);

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: padding,
      child: Text(text,
        style: TextStyle(
            color: MyColor().headerColor,
            fontFamily: 'Roboto-Black',
            fontSize: 14
        ),
      ),
    );
```

```
  }
}
```

Файл «bd.dart»

```
import 'dart:async';
import 'dart:io';
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
import 'package:reg_model/components/model/registration.dart';
import 'package:reg_model/components/citizenship.dart';


class DatabaseHelper {

  static final _databaseName = "reg_model.db";
  static final _databaseVersion = 1;

  // Make this a singleton class.
  DatabaseHelper._privateConstructor();

  static final DatabaseHelper instance =
DatabaseHelper._privateConstructor();

  // Only allow a single open connection to the database.
  static Database _database;

  Future<Database> get database async {
    if (_database != null) return _database;
    _database = await _initDatabase();
    return _database;
  }

  // open the database
  _initDatabase() async {
    String databasesPath = await getDatabasesPath();
    String path = join(databasesPath, _databaseName);
    // Open the database. Can also add an onUpdate callback
parameter.
    return await openDatabase(path,
        version: _databaseVersion,
        onCreate: this._create);
  }

  Future _create(Database db, int version) async {
    await db.execute(RegistrationModel().createTableQuery);
    await db.execute(Citizenship().createTableQuery);

    print("Created tables");
  }
```

```
  Future create() async {
    Database db = await database;
    this._create(db, _databaseVersion);
  }

  Future drop() async {
    try {
      Database db = await database;
      await db.execute("DROP TABLE IF EXISTS " +
RegistrationModel.tableName);
      await db.execute("DROP TABLE IF EXISTS " +
Citizenship.tableName);
      print('Dropped tables');
    } catch(e) {
      String error = 'Error dropping tables: ' + e.toString();
      print(error);
      return Future.error(error);
    }
  }

  // registration

  Future<RegistrationModel> upsertReg(RegistrationModel m) async
{
    Database db = await database;
    var count = Sqflite.firstIntValue(await db.rawQuery(
        "SELECT COUNT(*) FROM " + RegistrationModel.tableName));
    print('m count: $count');
    if (count == 0) {
      m.id = await db.insert(RegistrationModel.tableName,
m.toMap());
    } else {
      m.id = await db.update(
          RegistrationModel.tableName, m.toMap(), where: "id =
?",
          whereArgs: [m.id]);
    }

    return m;
  }

  Future<RegistrationModel> fetchReg([int id = 1]) async {
    Database db = await database;
    List<Map> results = await
db.query(RegistrationModel.tableName,
        columns: [
          'id',
          'lastName',
          'firstName',
```

```
          'middleName',
          'docIssuedBy',
          'docNum',
          'docSerial',
          'citizenship',
          'docType',
          'isMale',
          'noMiddleName',
          'birthDate',
          'docIssueDate',
          'photo',
          'email',
          'mobile',
          'cbNotifications',
          'cbRules'
        ],
        where: "id = ?",
        whereArgs: [id]);
    if (results.length > 0) {
      return RegistrationModel.fromMap(results.first);
    }
    return null;
  }

  Future<int> upsertCitizenship(List<Citizenship> cList) async {
    Database db = await database;
    int records = 0;
    await db.rawQuery("DELETE from ${Citizenship.tableName}");
    for (var c in cList) {
      c.id = await db.insert(Citizenship.tableName, c.toMap());
      records++;
    }
    return records;
  }

  Future<List<Citizenship>> fetchCitizenship([int id = 1]) async
{
    List<Citizenship> cList = new List();
    Database db = await database;
    List<Map> results = await db.query(Citizenship.tableName,
        columns: ['citizenshipTypeCode', 'description',
'engDescription'],
        where: "$id = ?",
        whereArgs: [id]);
    if (results.length > 0) {
      results.forEach((row) =>
cList.add(Citizenship.fromMap(row)));
      return cList;
    }
    return null;
```

```
  }
}
```

Файл «citizenship.dart»

```
class Citizenship {
  static final tableName = 'citizenship';
  int id = 1;

  String citizenshipTypeCode;
  String description;
  String engDescription;

  Citizenship({
    this.citizenshipTypeCode,
    this.description,
    this.engDescription
  });

  Citizenship.test(this.citizenshipTypeCode, this.description,
this.engDescription);

  String toString() {
    return engDescription;
  }

  factory Citizenship.fromJson(Map<String, dynamic> json) => new
Citizenship(
      citizenshipTypeCode: json["citizenshipTypeCode"],
      description: json["description"],
      engDescription: json["engDescription"]
  );

  Map<String, dynamic> toJson() => {
    "citizenshipTypeCode": citizenshipTypeCode,
    "description": description,
    "engDescription": engDescription
  };

  Map<String, dynamic> toMap() {
    return toJson();
  }

  static Citizenship fromMap(Map<String, dynamic> query) {
    Citizenship c = Citizenship();
    c.citizenshipTypeCode = query["citizenshipTypeCode"];
    c.description = query["description"];
    c.engDescription = query["engDescription"];
    return c;
  }
```

```
  String get createTableQuery =>
      "CREATE TABLE $tableName(id INTEGER PRIMARY KEY,"
          " citizenshipTypeCode TEXT,"
          " description TEXT,"
          " engDescription TEXT)";
}
```

Файл «citizenship_stream_builder.dart»

```
import 'dart:async';
import 'package:reg_model/components/bd.dart';
import 'package:reg_model/components/citizenship.dart';

class CitizenshipStreamBuilder {

  DatabaseHelper helper = DatabaseHelper.instance;
  List<Citizenship> model;

  void firstInit() async {
    model = await helper.fetchCitizenship();
    dataSink.add(model);
  }

  StreamController streamCitizenshipController =
StreamController<List<Citizenship>>.broadcast();

  Sink get dataSink => streamCitizenshipController.sink;
  Stream<List<Citizenship>> get dataStream =>
streamCitizenshipController.stream;
  void disposeStream(){
    streamCitizenshipController.close();
  }
  CitizenshipStreamBuilder() {
    dataSink.add(null);
  }
}
```

Файл «docType.dart»

```
class DocType {
  int id;
  String name;

  @override
  String toString() {
    return name;
  }

  DocType(this.id, this.name);
```

```
  DocType.fromJson(Map<String, dynamic> m) : this(m['id'],
m['name']);

  Map<String, dynamic> toJson() => {"id": id, "name": name};
}
toDocType(DocType docType){
  switch (docType.id){
    case 21:
      return docTypeRus[0];
    case 22:
      return docTypeRus[1];
    case 3:
      return docTypeRus[2];
    case 11:
      return docTypeEng[0];
  }
}

List<DocType> docTypeRus = [
  DocType(21, 'Russian passport'),
  DocType(22, 'Russian international passport'),
  DocType(3, 'Birth Certificates')
];

List<DocType> docTypeEng = [
  DocType(11, 'Passport')
];
```

Файл «photo.dart»

```
import 'dart:async';
import 'dart:convert';

import 'package:http/http.dart' as http;

class Ellips {
  final int x;
  final int y;
  final int width;
  final int height;
  final double a;
  final double b;
  final double e;
  final double c;

  Ellips(
      this.x, this.y, this.width, this.height, this.a, this.b,
this.e, this.c);
```

```
  Ellips.fromJson(Map<String, dynamic> json)
      : x = json['X'],
        y = json['Y'],
        width = json['Width'],
        height = json['Height'],
        a = json['a'],
        b = json['b'],
        e = json['e'],
        c = json['c'];
}

class Shapes {
  final List<String> points;

  Shapes(this.points);

  Shapes.fromJson(Map<String, dynamic> json)
      : points = new List<String>.from(json['Points']);
}

class Face {
  final List<String> points;

  Face(this.points);

  Face.fromJson(Map<String, dynamic> json)
      : points = new List<String>.from(json['Points']);
}

class Code {}

class FaceDetectResult {
  final Shapes shapes;
  final List<Face> faces;
  final String faceRectangle;
  final String neckRectangle;
  final Ellips ellips;
  final bool isSuccess;

  FaceDetectResult(this.shapes, this.faces, this.faceRectangle,
      this.neckRectangle, this.ellips, this.isSuccess);

  FaceDetectResult.fromJson(Map<String, dynamic> json)
      : shapes = Shapes.fromJson(json['Shapes']),
        faces = new List<Face>.from(
            (json['Faces'] as List<dynamic>).map((v) =>
Face.fromJson(v))),
        faceRectangle = json['FaceRectangle'],
        neckRectangle = json['NeckRectangle'],
        ellips = Ellips.fromJson(json['Ellips']),
```

```
        isSuccess = json['IsSuccess'];
}

class WaitValue {
  final double minValue;
  final double maxValue;
  final double target;
  final double weight;

  WaitValue(this.minValue, this.maxValue, this.target,
this.weight);

  WaitValue.fromJson(Map<String, dynamic> json)
      : minValue = json['MinValue'],
        maxValue = json['MaxValue'],
        target = json['Target'],
        weight = json['Weight'];
}

class ValidationList {
  final int type;
  final bool isFatalError;
  final bool hasError;
  final WaitValue waitValue;
  final double realValue;
  final int aligment;
  final String name;

  ValidationList(this.type, this.isFatalError, this.hasError,
this.waitValue,
      this.realValue, this.aligment, this.name);

  ValidationList.fromJson(Map<String, dynamic> json)
      : type = json['Type'],
        isFatalError = json['IsFatalError'],
        hasError = json['HasError'],
        waitValue = WaitValue.fromJson(json['WaitValue']),
        realValue = json['RealValue'],
        aligment = json['Aligment'],
        name = json['Name'];
}

class ValidationResult {
  final int code;
  final String log;
  final int points;
  final int contrast;
  final int faceFound;
  final int faceIllumination;
  final int faceSize;
```

```
  final int focus;
  final int glare;
  final int headTilt;
  final int headTurning;
  final int positioning;

  ValidationResult(
      this.code,
      this.log,
      this.points,
      this.contrast,
      this.faceFound,
      this.faceIllumination,
      this.faceSize,
      this.focus,
      this.glare,
      this.headTilt,
      this.headTurning,
      this.positioning);

  ValidationResult.fromJson(Map<String, dynamic> j)
      : code = j['Code'],
        log = j['Log'],
        points = j['Points'],
        contrast = j['Contrast'],
        faceFound = j['FaceFound'],
        faceIllumination = j['FaceIllumination'],
        faceSize = j['FaceSize'],
        focus = j['Focus'],
        glare = j['Glare'],
        headTilt = j['HeadTilt'],
        headTurning = j['HeadTurning'],
        positioning = j['Positioning'];
}

class FaceCheckResult {
  final String faceRectangle;
  final Ellips ellips;
  final Shapes shapes;
  final String pictureRectangle;
  final List<ValidationList> validationList;
  final ValidationResult validationResult;

  FaceCheckResult(this.faceRectangle, this.ellips, this.shapes,
      this.pictureRectangle, this.validationList,
this.validationResult);

  FaceCheckResult.fromJson(Map<String, dynamic> j)
      : faceRectangle = j['FaceRectangle'],
        ellips = j['Ellips'] == null ? null :
```

```
Ellips.fromJson(j['Ellips']),
        shapes = j['Shapes'] == null ? null :
Shapes.fromJson(j['Shapes']),
        pictureRectangle = j['PictureRectangle'],
        validationList = j['ValidationList'] == null
            ? null
            : new List<ValidationList>.from(
            (j['ValidationList'] as List<dynamic>)
                .map((f) => ValidationList.fromJson(f))),
        validationResult = j['ValidationResult'] == null
            ? null
            : ValidationResult.fromJson(j['ValidationResult']);
}

enum Checks {
  MinImageWidth,
  MaxImageWidth,
  MinImageHeigth,
  RatioBetweenEyes,
  RatioDistanceFromLeftBorderEye,
  DistanceEyesFromTopBorder,
  RatioHeightToWidth,
  BadImageFormat,
  CannotRecognize,
  RatioDistanceFromRigthBorderEyes,
  RatioFaceHeight,
  RatioFaceWidth,
  FaceNotFound,
  HeadTurning,
  HeadTilt,
  RatioDistanceFromTopBorderEyesMin,
  Background,
  FaceOutOfBorder,
  FaceIllumination,
  FaceSquarePercent,
  BackgroundHistogramThreshold,
  FaceHistogramThreshold,
  BackgroundDispersion,
  BackgroundIncorrectColor,
  FaceSimilarWithBackgroundColor,
  FaceDispersion,
  FaceIncorrectColor,
  FaceMassDelta,
  MaybeHairOutOfBorder,
  MaybeHairOutOfBorderTop,
  FaceGrayColor,
  MouthOpen,
  RigthEyeClosed,
  LeftEyeClosed
}
```

```
class Failure {
  Checks type;
  String desc;

  Failure(this.type, this.desc);
}

class PhotoValidationResult {
  int x, y, width, height;
  List<Failure> failures;
}

class PhotoCheckerV2 {
  static const FACE_DETECT_URL =
"/api/v1/facedetector/isoface/facedetect";
  static const FACE_CHECK_URL =
"/api/v1/facedetector/isoface/full";

  String serviceURL;

  String getServiceURL() {
    return serviceURL;
  }

  void setServiceURL(String serviceURL) {
    if (serviceURL.endsWith("/"))
      serviceURL = serviceURL.substring(0, serviceURL.length -
1);
    this.serviceURL = serviceURL;
  }

  Future<Map<String, dynamic>> fetchPost(String url, dynamic
body) async {
    for (int n = 0; n < 4; ++n) {
      final response = await http.post(Uri.encodeFull(url),
body: body);

      if (response.statusCode == 200) {
        return json.decode(response.body);
      } else if (n >= 4) {
        throw 'Post to service failed';
      }
    }
  }
  Future<FaceDetectResult> detectFace(image) async {
    return fetchPost(serviceURL + FACE_DETECT_URL, image)
        .then((v) => FaceDetectResult.fromJson(v));
  }
```

```
  Future<FaceCheckResult> executeFaceCheck(image) {
    return fetchPost(
        serviceURL + FACE_CHECK_URL +
"?crop=$enableFaceCropOnValidate",
        image)
        .then((v) => FaceCheckResult.fromJson(v));
  }

  bool enableFaceCropOnValidate = true;

  bool isEnableFaceCropOnValidate() {
    return enableFaceCropOnValidate;
  }

  void setEnableFaceCropOnValidate(bool
enableFaceCropOnValidate) {
    this.enableFaceCropOnValidate = enableFaceCropOnValidate;
  }

  Future<PhotoValidationResult> validatePhoto(image) async {
    PhotoValidationResult validationResult = new
PhotoValidationResult();
    FaceCheckResult res = await executeFaceCheck(image);
    if (res.pictureRectangle != null) {
      List<String> pictureCoords =
res.pictureRectangle.split(",");
      if (pictureCoords.length >= 4) {
        validationResult.x = int.parse(pictureCoords[0].trim());
        validationResult.y = int.parse(pictureCoords[1].trim());
        validationResult.width =
int.parse(pictureCoords[2].trim());
        validationResult.height =
int.parse(pictureCoords[3].trim());
      }
    }
    List<Failure> fatalFails = [];
    if (res.validationList==null) {
      //throw
allTranskations.text("error_validator_invalid_image");
    }
    for (ValidationList v in res.validationList) {
      if (v.isFatalError) {
        if (fatalFails == null) {
          fatalFails = new List<Failure>();
        }
        Checks check =
        Checks.values.firstWhere((e) => e.toString() ==
'Checks.' + v.name);
        if (check == null) continue;
        String desc =
```

```
        getFailDescription(check, v.realValue >
v.waitValue.maxValue);
        if (desc != null) {
          bool duplicate = false;
          for (Failure f in fatalFails) {
            if (f.desc == desc) {
              duplicate = true;
              break;
            }
          }
          if (!duplicate) {
            fatalFails.add(new Failure(check, desc));
          }
        }
      }
    }
    validationResult.failures = fatalFails;
    return validationResult;
  }
  String getFailDescription(Checks check, bool maxLimit) {
    switch (check) {
      case Checks.RatioBetweenEyes:
        {
          if (maxLimit)
            return "Distance between the pupils is too large";
          else
            return "Distance between the pupils is too small";
        }
        break;
      case Checks.RatioDistanceFromLeftBorderEye:
        {
          if (maxLimit)
            return "Face too far from the left border";
          else
            return "Face too close to the left border";
        }
        break;
      case Checks.RatioDistanceFromTopBorderEyesMin:
      case Checks.DistanceEyesFromTopBorder:
        {
          if (maxLimit)
            return "Face too far from the upper border";
          else
            return "Face too close to the upper border";
        }
        break;
      case Checks.RatioHeightToWidth:
        {
          return "Invalid facial proportions (height/width)";
        }
```

```
        break;
      case Checks.BadImageFormat:
        return "Bad image format";
      case Checks.FaceNotFound:
      case Checks.CannotRecognize:
        return "Face not found";
      case Checks.RatioDistanceFromRigthBorderEyes:
        {
          if (maxLimit)
            return "Face too far from the right border";
          else
            return "Face too close to the right border";
        }
        break;
      case Checks.RatioFaceHeight:
        {
          if (maxLimit)
            return "Vertical size of the face is too large";
          else
            return "Too small vertical size of the face";
        }
        break;
      case Checks.RatioFaceWidth:
        {
          if (maxLimit)
            return "Horizontal size of the face is too large";
          else
            return "Too small horizontal size of the face";
        }
        break;
      case Checks.HeadTurning:
        return "Excessive head rotation";
      case Checks.HeadTilt:
        return "Excessive head tilt";
      case Checks.Background:
      case Checks.BackgroundIncorrectColor:
        return "Invalid background color (light background
required)";
      case Checks.FaceOutOfBorder:
      case Checks.MaybeHairOutOfBorder:
      case Checks.MaybeHairOutOfBorderTop:
        return "Face out of border";
      case Checks.FaceMassDelta:
      case Checks.FaceIncorrectColor:
      case Checks.FaceIllumination:
        return "The colors in the picture are incorrectly
displayed";
      case Checks.FaceSquarePercent:
        return null;
      case Checks.BackgroundHistogramThreshold:
```

```
        return "Too dark background";
      case Checks.FaceDispersion:
      case Checks.FaceHistogramThreshold:
        return "The face is out of focus";
      case Checks.BackgroundDispersion:
        return "Background is too patchy";
      case Checks.FaceSimilarWithBackgroundColor:
        return "Background merges with the face";
      case Checks.MouthOpen:
        return "The mouth is open";
      case Checks.LeftEyeClosed:
        return "The left eye is closed";
      case Checks.RigthEyeClosed:
        return "The right eye is closed";
      case Checks.FaceGrayColor:
        return "The photo is black and white";
      default:
        {
          return null;
        }
    }
  }
}
```

Файл «sharedpreferences.dart»

```
import 'dart:async';
import 'package:shared_preferences/shared_preferences.dart';

class SharedPrefs {
  static SharedPreferences storage;
  static const String _storageKey = "FanIdApplication_";

  static Future<String> get(String name) async {
    SharedPreferences prefs = await
SharedPreferences.getInstance();
    return prefs.getString(_storageKey + name) ?? '';
  }

  static set(String name, String value) async {
    SharedPreferences prefs = await
SharedPreferences.getInstance();
    prefs.setString(_storageKey + name, value);
  }

  static Future clear() async {
    try {
      SharedPreferences prefs = await
SharedPreferences.getInstance();
      prefs.clear();
```

```
      print('SharedPrefs cleared');
    } catch(e) {
      String error = 'Error clearing SharedPrefs: ' +
e.toString();
      print(error);
      return Future.error(error);
    }
  }
}
```

Файл «user_data_stream_builder.dart»

```
import 'dart:async';

import 'package:reg_model/components/bd.dart';
import 'package:reg_model/components/model/registration.dart';

class UserDataStreamBuilder {

  DatabaseHelper helper = DatabaseHelper.instance;
  RegistrationModel model;

  void firstInit() async {
    await helper.fetchReg().then((value){
      model = value;
      dataSink.add(model);
    }).catchError((er){
      print(er.toString());
    });
  }
  StreamController streamUserDataController =
  StreamController<RegistrationModel>.broadcast();

  Sink get dataSink => streamUserDataController.sink;

  Stream<RegistrationModel> get dataStream =>
streamUserDataController.stream;

  void disposeStream(){
    streamUserDataController.close();
  }

  UserDataStreamBuilder(){
    dataSink.add(null);
  }
}
```

Файл «validator.dart»

```
import 'package:reg_model/step.dart';

class Validators {

  static String notEmpty(String txt) {
    if (txt == null || txt.isEmpty) {
      return "Please enter a value";
    }
    return null;
  }

  static String latinName(String txt) {
    final RegExp regExp = new RegExp(r"^[a-zA-Z \-.`]*$");
    if (txt.startsWith("-") || txt.endsWith("-")) {
      return "Unresolved character \"-\"";
    }
    if (!regExp.hasMatch(txt)) {
      return "Only Latin letters and delimiters are allowed";
    }
    return notEmpty(txt);
  }

  static String serial(String value, int y) {
    value = value.replaceAll(new RegExp(r"[^0-9]"), "");
    String e = notEmpty(value);
    if (e != null) return e;
    final Pattern pattern = (y == 4 ? r'^[0-9]{4}$' : r'^[0-
9]{2}$');
    RegExp regex = new RegExp(pattern);
    if (!regex.hasMatch(value))
      return "Enter a valid document series";
    else
      return null;
  }

  static String number(String value, int y) {
    value = value.replaceAll(new RegExp(r"[^0-9]"), "");
    String e = notEmpty(value);
    if (e != null) return e;
    final Pattern pattern = (y == 6 ? r'^[0-9]{6}$' : r'^[0-
9]{7}$');
    RegExp regex = new RegExp(pattern);
    if (!regex.hasMatch(value))
      return "Enter valid document number";
    else
      return null;
  }

  static String seralSR(String value) {
    String e = notEmpty(value);
```

```
    if (e != null) return e;
    final Pattern pattern = r'^[0-9a-zA-Za-яA-ЯёЁйЙ\-]{3,9}$';
    RegExp regex = new RegExp(pattern);
    if (!regex.hasMatch(value))
      return "Enter a valid document series";
    else
      return null;
  }

  static String isDate(String value, DateTime minDate, DateTime
maxDate) {
    try {
      DateTime date = toDateTime(value);
      if (minDate != null && minDate.compareTo(date) > 0) {
        return "Date less than acceptable";
      }
      if (maxDate != null && maxDate.compareTo(date) < 0) {
        return "Date is more than acceptable";
      }
      return null;
    } catch (e) {
      return "Incorrect date";
    }
  }

  static String email(String value) {
    String e = notEmpty(value);
    if (e != null) return e;
    final Pattern pattern =

r'^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\")
)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-
Z\-0-9]+\.)+[a-zA-Z]{2,}))$';
    RegExp regex = new RegExp(pattern);
    if (!regex.hasMatch(value))
      return "Enter valid e-mail";
    else
      return null;
  }


  static String phone(String value) {
    value = value.replaceAll(new RegExp(r"[^0-9]"), "");
    String e = notEmpty(value);
    if (e != null) return e;
    final Pattern patternIntl =
        r'^[0-9]{4,16}$';
    final Pattern patternRus =
        r'^[0-9]{11}$';
    RegExp regex;
```

```
    if (value.startsWith("7"))
      regex = new RegExp(patternRus);
    else
      regex = new RegExp(patternIntl);
    if (!regex.hasMatch(value))
      return "Enter a valid phone number";
    else
      return null;
  }

  static String code(String value) {
    value = value.replaceAll(new RegExp(r"[^0-9]"), "");
    String e = notEmpty(value);
    if (e != null) return e;
    final Pattern pattern = r'^[0-9]{6}$';
    RegExp regex = new RegExp(pattern);
    if (!regex.hasMatch(value))
      return "Enter valid code";
    else
      return null;
  }
}
```

Файл «registration.dart»

```
import 'dart:convert';
import 'package:scoped_model/scoped_model.dart';
import 'package:reg_model/step.dart';
import 'package:reg_model/components/docType.dart';
import 'package:reg_model/components/citizenship.dart';

class RegistrationModel extends Model {
  static final tableName = 'reg';
  int id = 1;

  bool cbNotifications = false;
  bool cbRules = false;
  List<int> photo;
  bool isMale;
  String lastName, firstName, middleName;
  bool noMiddleName = false;
  DateTime birthDate;
  Citizenship citizenship;
  DocType docType;
  String docNum, docSerial, docIssuedBy;
  DateTime docIssueDate;
  String email, mobile;
  bool notRusMobile = false;

  RegistrationModel(){}
```

```
  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'lastName': lastName,
      'firstName': firstName,
      'middleName': middleName,
      'docIssuedBy': docIssuedBy,
      'docNum': docNum,
      'docSerial': docSerial,
      'citizenship': citizenship == null ? null :
jsonEncode(citizenship.toJson()),
      'docType': docType == null ? null :
jsonEncode(docType.toJson()),
      'isMale': fromBool(isMale),
      'noMiddleName': fromBool(noMiddleName),
      'birthDate': birthDate == null ? null :
toString(birthDate),
      'docIssueDate': docIssueDate == null ? null :
toString(docIssueDate),
      'photo': photo,
      'email': email,
      'mobile': mobile,
      'cbNotifications': fromBool(cbNotifications),
      'cbRules': fromBool(cbRules)
    };
  }

  int fromBool(bool value){
    switch (value){
      case true:
        return 1;
      case false:
        return 0;
      default:
        return 2;
    }
  }

  static RegistrationModel fromMap(Map<String, dynamic> query) {
    RegistrationModel model = RegistrationModel();
    model.id = query['id'];
    model.lastName = query['lastName'];
    model.firstName = query['firstName'];
    model.middleName = query['middleName'];
    model.docIssuedBy = query['docIssuedBy'];
    model.docNum = query['docNum'];
    model.docSerial = query['docSerial'];
    model.citizenship = (query['citizenship'] == null ? null :
Citizenship.fromJson(jsonDecode(query['citizenship'])));
```

```
    model.docType = query['docType'] == null ? null :
toDocType(DocType.fromJson(jsonDecode(query['docType'])));
    model.isMale = toBool(query['isMale']);
    model.noMiddleName = toBool(query['noMiddleName']);
    model.birthDate = (query['birthDate'] == null ? null :
toDateTime(query['birthDate']));
    model.docIssueDate = (query['docIssueDate'] == null ? null :
toDateTime(query['docIssueDate']));
    model.photo = query['photo'];
    model.email = query['email'];
    model.mobile = query['mobile'];
    model.cbNotifications = toBool(query['cbNotifications']);
    model.cbRules = toBool(query['cbRules']);
    return model;
  }

  static toBool(int value){
    switch (value){
      case 1:
        return true;
      case 0:
        return false;
      default:
        return null;
    }
  }

  void fromJson(Map<String, dynamic> m) {
    id = m['id'];
    lastName = m['lastName'];
    firstName = m['firstName'];
    middleName = m['middleName'];
    isMale = m['isMale'];
    noMiddleName=m['noMiddleName'];
    birthDate = toDateTime(m['birthDate']);
    citizenship = Citizenship.fromJson(m['citizenship']);
    docType = DocType.fromJson(m['docType']);
    docSerial = m['docSerial'];
    docNum = m['docNum'];
    docIssuedBy=m['docIssuedBy'];
    docIssueDate = toDateTime(m['docIssueDate']);
    photo = m['photo'];
    email = m['email'];
    mobile = m['mobile'];
    cbNotifications = m['cbNotifications'];
    cbRules = m['cbRules'];
  }

  Map<String, dynamic> toJson() => {
    'id': id,
```

```
    'lastName': lastName,
    'firstName': firstName,
    'middleName': middleName,
    'isMale': isMale,
    'noMiddleName':noMiddleName,
    'birthDate': birthDate == null ? null : toString(birthDate),
    'citizenship': citizenship == null ? null :
citizenship.toJson(),
    'docType': docType == null ? null : docType.toJson(),
    'docSerial': docSerial,
    'docNum': docNum,
    'docIssuedBy':docIssuedBy,
    'docIssueDate':docIssueDate == null ? null :
toString(docIssueDate),
    'photo':photo,
    'email': email,
    'mobile': mobile,
    'cbNotifications': cbNotifications,
    'cbRules': cbRules,
  };

  String get createTableQuery =>
      "CREATE TABLE $tableName(id INTEGER,"
          " lastName TEXT,"
          " firstName TEXT,"
          " middleName TEXT,"
          " docIssuedBy TEXT,"
          " docNum TEXT,"
          " docSerial TEXT,"
          " citizenship TEXT,"
          " docType TEXT,"
          " isMale INTEGER,"
          " noMiddleName INTEGER,"
          " birthDate TEXT,"
          " docIssueDate TEXT,"
          " photo BLOB,"
          " email TEXT,"
          " mobile TEXT,"
          " cbNotifications INTEGER,"
          " cbRules INTEGER)";
}
```

Файл «dict_request.dart»

```
import "dart:convert";
import
'package:reg_model/components/network/request/core_request.dart'
;
import
'package:reg_model/components/network/request/encoder_request.da
```

```
rt';
import 'package:reg_model/components/sharedpreferences.dart';

class DictRequest extends CoreRequest {
  String instanceGuid;

  DictRequest();

  loadGuidAndToken() async {
    this.instanceGuid = await SharedPrefs.get('instanceGuid');
    this.token = await SharedPrefs.get('token');
  }

  Map<String, dynamic> toJson() =>
      {
        'instanceGuid': instanceGuid,
        'sign': new RequestEncoder().encodeDict(this).sign,
      };

  String toJsonString() {
    return jsonEncode(this);
  }
}
```

Файл «encoder_request.dart»

```
import "dart:convert";
import "package:crypto/crypto.dart";
import
'package:reg_model/components/network/request/instance_request.d
art';
import
'package:reg_model/components/network/request/dict_request.dart'
;
import
'package:reg_model/components/network/request/otp_request.dart';
import
'package:reg_model/components/network/request/new_request.dart';

class RequestEncoder {
  InstanceRequest encodeInstance(InstanceRequest r) {
    String result;
    List<int> encoded;
    result =  r.instanceGuid +
        r.appVersion.toString() +
        r.platform.toString() +
        r.deviceId +
        r.deviceInfo +
        r.token;
    encoded = utf8.encode(result);
```

```
    r.unSigned = String.fromCharCodes(encoded); // test
    r.sign = sha256.convert(encoded).toString();
    return r;
  }

  DictRequest encodeDict(DictRequest r) {
    String result;
    List<int> encoded;
    result = r.instanceGuid +
        r.token;
    encoded = utf8.encode(result);
    r.sign = sha256.convert(encoded).toString();
    return r;
  }

  OtpRequest encodeOtp(OtpRequest r) {
    String result;
    List<int> encoded;
    result =  r.instanceGuid +
        r.mobile +
        r.token;
    encoded = utf8.encode(result);
    r.sign = sha256.convert(encoded).toString();
    return r;
  }

  NewRequest encodeNew(NewRequest r) {
    String result;
    List<int> encoded;

    result = r.lastName +
        r.firstName +
        r.middleName +
        r.birthDate +
        r.citizenshipCode +
        r.email +
        r.otpRqId.toString() +
        r.otp +
        r.idType.toString() +
        r.idNumber +
        r.idIssuer +
        r.idIssueDate +
        r.photo +
        r.gender.toString() +
        r.token;
    encoded = utf8.encode(result);
    r.sign = sha256.convert(encoded).toString();
    return r;
  }
}
```

Файл «new_request.dart»

```
import "dart:convert";
import
'package:reg_model/components/network/request/core_request.dart'
;
import
'package:reg_model/components/network/request/encoder_request.da
rt';
import 'package:reg_model/components/sharedpreferences.dart';

class NewRequest extends CoreRequest {
  String lastName;
  String firstName;
  String middleName;
  String birthDate;
  String citizenshipCode;
  String email;
  int otpRqId;
  String otp;
  int idType;
  String idNumber;
  String idIssuer;
  String idIssueDate;
  String photo;
  int gender;

  NewRequest(this.lastName, this.firstName, this.middleName,
this.birthDate,
      this.citizenshipCode, this.email, this.otpRqId, this.otp,
this.idType,
      this.idNumber, this.idIssuer, this.idIssueDate,
this.photo, this.gender);

  loadGuidAndToken() async {
    this.instanceGuid = await SharedPrefs.get('instanceGuid');
    this.token = await SharedPrefs.get('token');
  }

  Map<String, dynamic> toJson() =>
      {
        'lastName': lastName,
        'firstName': firstName,
        'middleName': middleName,
        'birthDate': birthDate,
        'citizenshipCode': citizenshipCode,
        'email': email,
        'otpRqId': otpRqId,
        'otp': otp,
        'idType': idType,
```

```
        'idNumber': idNumber,
        'idIssuer': idIssuer,
        'idIssueDate': idIssueDate,
        'photo': photo,
        'gender': gender,
      };

  String getSign() {
    RequestEncoder r = new RequestEncoder();
    NewRequest request = r.encodeNew(this);
    return request.sign;
  }

  String toJsonString() {
    return jsonEncode(this);
  }

}
```

Файл «otp_request.dart»

```
import "dart:convert";
import
'package:reg_model/components/network/request/core_request.dart'
;
import
'package:reg_model/components/network/request/encoder_request.da
rt';
import 'package:reg_model/components/sharedpreferences.dart';

class OtpRequest extends CoreRequest {
  //String instanceGuid;
  String mobile;

  OtpRequest(this.mobile);

  loadGuidAndToken() async {
    this.instanceGuid = await SharedPrefs.get('instanceGuid');
    this.token = await SharedPrefs.get('token');
  }

  Map<String, dynamic> toJson() =>
      {
        'instanceGuid': instanceGuid,
        'mobile': mobile,
        'sign': new RequestEncoder().encodeOtp(this).sign,
      };

  String toJsonString() {
    return jsonEncode(this);
```

```
  }
}
```

Файл «dict_response.dart»

```
import 'dart:convert';
import 'package:reg_model/components/citizenship.dart';

class DictResponse {
  int resultCode;
  String desc;
  List<Citizenship> citizenshipTypes;

  DictResponse({
    this.resultCode,
    this.desc,
    this.citizenshipTypes
  });

  String toString() {
    return jsonEncode(this);
  }

  factory DictResponse.fromJson(Map<String, dynamic> json) =>
new DictResponse(
      resultCode: json["resultCode"],
      desc: json["desc"],
      citizenshipTypes: (json["citizenshipTypes"] as
List).map((i) => Citizenship.fromJson(i)).toList()
  );

  Map<String, dynamic> toJson() => {
    "resultCode": resultCode,
    "desc": desc,
    "citizenshipTypes": citizenshipTypes,
  };
}
```

Файл «inctance_response.dart»

```
class InstanceResponse {
  int resultCode;
  String desc;
  String instanceGuid;
  String token;

  InstanceResponse({
    this.resultCode,
    this.desc,
    this.instanceGuid,
```

```
    this.token,
  });

  factory InstanceResponse.fromJson(Map<String, dynamic> json)
=> new InstanceResponse(
    resultCode: json["resultCode"],
    desc: json["desc"],
    instanceGuid: json["instanceGuid"],
    token: json["token"],
  );

  Map<String, dynamic> toJson() => {
    "resultCode": resultCode,
    "desc": desc,
    "instanceGuid": instanceGuid,
    "token": token,
  };
}
```

Файл «new_response.dart»

```
class NewResponse {
  int resultCode;
  String desc;
  int registrationId;

  NewResponse({
    this.resultCode,
    this.desc,
    this.registrationId
  });

  factory NewResponse.fromJson(Map<String, dynamic> json) => new
NewResponse(
      resultCode: json["resultCode"],
      desc: json["desc"],
      registrationId: json["registrationId"]
  );

  Map<String, dynamic> toJson() => {
    "resultCode": resultCode,
    "desc": desc,
    "registrationId": registrationId
  };
}
```

Файл «otp_response.dart»

```
class OtpResponse {
  int resultCode;
```

```
  String desc;
  int otpRqId;

  OtpResponse({
    this.resultCode,
    this.desc,
    this.otpRqId
  });

  factory OtpResponse.fromJson(Map<String, dynamic> json) => new
OtpResponse(
      resultCode: json["resultCode"],
      desc: json["desc"],
      otpRqId: json["otpRqId"]
  );

  Map<String, dynamic> toJson() => {
    "resultCode": resultCode,
    "desc": desc,
    "otpRqId": otpRqId
  };
}
```

Файл «dict.dart»

```
import 'dart:async';
import 'dart:convert';
import 'dart:io';

import 'package:http/http.dart' as http;
import 'package:reg_model/step.dart';
import 'package:reg_model/components/bd.dart';
import
'package:reg_model/components/network/response/dict_response.dar
t';
import
'package:reg_model/components/network/request/dict_request.dart'
;
import
'package:reg_model/components/network/response/result_code.dart'
;

class Dict {

  final String mc = "/dict";

  Future<DictResponse> fetchDict() async {

    try {
      final response = await http.post(url + mc,
```

```
          headers: {
            HttpHeaders.contentTypeHeader: 'application/json'
          },
          body: await requestBody()
      );

      if (response.statusCode == 200) {
        return
DictResponse.fromJson(json.decode(response.body));
      } else {
        String error = 'Server error ' +
response.statusCode.toString() + ' ' +
            response.reasonPhrase + "\nbody: " + response.body;
        print(error);
        return Future.error(error);
      }
    } catch(e) {
      String error = 'mcdict: ' + e.toString();
      print(error);
      return Future.error(error);
    }
  }

  Future<String> requestBody() async {

    DictRequest r = DictRequest();
    await r.loadGuidAndToken();
    String json = r.toJsonString();
    print("Request: " + json);
    return json;
  }


  String saveResponseData(DictResponse data) {
    String message;
    switch(data.resultCode) {
      case ResultCode.RC_OK: {
        DatabaseHelper helper = DatabaseHelper.instance;

helper.upsertCitizenship(data.citizenshipTypes).then((value) {
          var records = value;
          print('inserted citizenships: $records');
        }).catchError((error){
          print(error.toString());
          return error.toString();
        });
        break;
      }
      case ResultCode.RC_ALREADY_REGISTERED: {
        break;
```

```
      }
    }
    print("Response: " + data.toJson().toString());
    message = "Server message: " + data.toJson().toString() +
"\n\n" + ResultCode.desc(data.resultCode);
    return message;
  }

}
```

Файл «new.dart»

```
import 'dart:async';
import 'dart:convert';
import 'dart:io';

import 'package:http/http.dart' as http;
import 'package:reg_model/step.dart';
import
'package:reg_model/components/network/response/new_response.dart
';
import
'package:reg_model/components/network/request/new_request.dart';
import
'package:reg_model/components/network/response/result_code.dart'
;

class New {
  String lastName;
  String firstName;
  String middleName;
  String birthDate;
  String citizenshipCode;
  String email;
  int otpRqId;
  String otp;
  int idType;
  String idNumber;
  String idIssuer;
  String idIssueDate;
  String photo;
  int gender;

  New(this.lastName, this.firstName, this.middleName,
this.birthDate,
      this.citizenshipCode, this.email,
      this.otpRqId,
      this.otp,
      this.idType, this.idNumber, this.idIssuer,
this.idIssueDate,
```

```
        this.photo,
        this.gender);
    New.noargs();

    final String mc = "/new";
    NewRequest newRequest;

    Future<NewResponse> fetchNew() async {

      try {
        final response = await http.post(url + mc ,
            headers: {
              HttpHeaders.contentTypeHeader: 'application/json'
            },
            body: await requestBody()
        );

        if (response.statusCode == 200) {
          return NewResponse.fromJson(json.decode(response.body));
        } else {
          String error = 'Server error ' +
  response.statusCode.toString() + ' ' +
              response.reasonPhrase + "\nbody: " + response.body;
          print(error);
          return Future.error(error);
        }
      } catch(e) {
        String error = 'mcnew: ' + e.toString();
        print(error);
        return Future.error(error);
      }
    }

    Future<String> requestBody() async {
      NewRequest r = NewRequest(
          lastName,
          firstName,
          middleName,
          birthDate,
          citizenshipCode,
          email,
          otpRqId,
          otp,
          idType,
          idNumber,
          idIssuer,
          idIssueDate,
          photo,
          gender,
      );
```

```
    await r.loadGuidAndToken();
    String json = r.toJsonString();
    print("Request: " + json);
    String gh=r.gender.toString();
    print(gh);
    return json;
  }


  String saveResponseData(NewResponse data) {
    String message;
    switch(data.resultCode) {
      case ResultCode.RC_OK: {
        break;
      }
      case ResultCode.RC_ALREADY_REGISTERED: {
        break;
      }
    }
    print("Response: " + data.toJson().toString());
    message = "Server message: " + data.toJson().toString() +
"\n\n" +
        ResultCode.desc(data.resultCode);
    return message;
  }
}
```

Файл «otp.dart»

```
import 'dart:async';
import 'dart:convert';
import 'dart:io';

import 'package:http/http.dart' as http;
import 'package:reg_model/step.dart';
import
'package:reg_model/components/network/response/otp_response.dart
';
import
'package:reg_model/components/network/request/otp_request.dart';
import
'package:reg_model/components/network/response/result_code.dart'
;

class Otp {

  final String mc = "/otp";
  String mobile;
```

```
  Otp(this.mobile);
  Otp.noargs();

  Future<OtpResponse> fetchOtp() async {

    try {
      final response = await http.post(url + mc,
          headers: {
            HttpHeaders.contentTypeHeader: 'application/json',
            HttpHeaders.authorizationHeader: 'Basic api_token'
          },
          body: await requestBody()
      );

      if (response.statusCode == 200) {
        // If the call to the server was successful, parse the
JSON
        return OtpResponse.fromJson(json.decode(response.body));
      } else {
        String error = 'Server error ' +
response.statusCode.toString() + ' ' +
            response.reasonPhrase + "\nbody: " + response.body;
        print(error);
        return Future.error(error);
      }
    } catch(e) {
      String error = 'mcotp: ' + e.toString();
      print(error);
      return Future.error(error);
    }
  }

  Future<String> requestBody() async {
    OtpRequest r = OtpRequest(mobile);
    await r.loadGuidAndToken();
    String json = r.toJsonString();
    print("Request: " + json);
    return json;
  }


  String saveResponseData(OtpResponse data) {
    String message;
    switch(data.resultCode) {
      case ResultCode.RC_OK: {
        break;
      }
      case ResultCode.RC_ALREADY_REGISTERED: {
        break;
      }
```

```
    }
    print("Response: " + data.toJson().toString());
    message = "Server message: " + data.toJson().toString() +
"\n\n" +
        ResultCode.desc(data.resultCode);
    return message;
  }
}
```

Приложение В. Описание REST API для работы с пользователями

Регистрация нового пользователя.

Метод: *POST*

URI:https://68f5f049-374a-4665-8bbf-898124a6af24.mock.pstmn.io/user/new

Тело запроса:

```
{
  "lastName": "String",
  "firstName": "String",
  "middleName": "String",
  "birthDate": "String",
  "citizenshipCode": "String",
  "email": "String",
  "otpRqId": int,
  "otp": "String",
  "idType": int,
  "idNumber": "String",
  "idIssuer": "String",
  "idIssueDate": "String",
  "photo": "String",
  "gender ": int
}
```

Ответ:

200 OK

```
{
  "resultCode": int,
  "desc": "String",
  "registrationId": int
}
```

400 Bed Request

```
{
  "desc": "String"
}
```

Получение спрвочника

Метод: *POST*

URI:https://68f5f049-374a-4665-8bbf-898124a6af24.mock.pstmn.io/dict

Тело запроса:

```
{
  "instanceGuid": "String",
  "sign": "String"
}
```

Ответ:

200 OK

```
{
  "resultCode": int,
  "desc": "String",
  "citizenshipTypes": [
    {
      "citizenshipTypeCode":"String",
      "description":"String",
      "engDescription":"String"
    }
  ]
}
```

400 Bed Request

```
{
  "desc": "String"
}
```

Отправка одноразового кода регистрации

Метод: *POST*

URI:https://68f5f049-374a-4665-8bbf-898124a6af24.mock.pstmn.io/user/otp

Тело запроса:

```
{
  "instanceGuid": "String",
  "mobile": "String",
  "sign": "String"
}
```

Ответ:

200 OK

```
{
  "resultCode": int,
  "desc": "String",
  "otpRqId": int
}
```

400 Bed Request

```
{
  "desc": "String"
}
```

Приложение С. Презентация

Библиотека регистрации и идентификации пользователя для мобильного клиента

Автор работы: студентка группы 16ВВ1 Забродина К.А
Научный руководитель: к.т.н., доцент кафедры ВТ Токарев А.Н.

Пенза, 2020

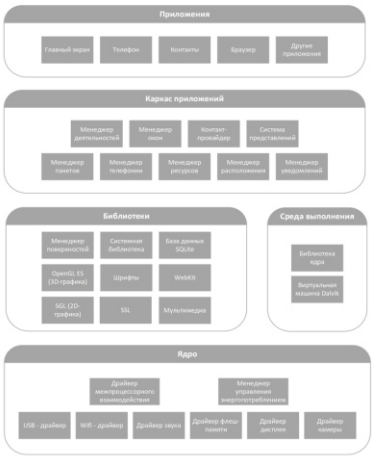
Цель и объект работы

Цель работы: разработать библиотеку регистрации и идентификации пользователя для мобильного клиента.

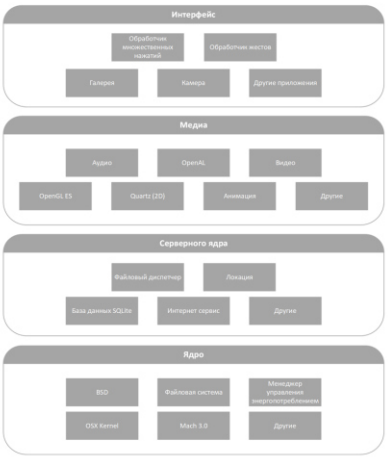
Объектом разработки является библиотека, обеспечивающая выполнение следующих функций: сбор, хранение и обработку персональных данных пользователя.

Хранение информации о пользователях, организациях и группах должно осуществляться в БД. Необходимо реализовать механизмы асинхронной обработки запросов от сторонних сервисов. Также требуется реализовать приложение, демонстрирующее функционал библиотеки.

Мобильные операционные системы



Архитектура ОС Android



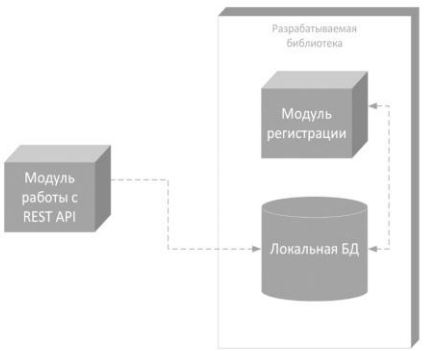
Архитектура ОС iOS

Стек технологий

Flutter Dart Android Studio™

Дополнительные инструменты: shared_preferences, http, sqflite, scoped_model, image_picker, url_launcher.

Структура проекта



Структура приложения

Взаимодействия компонентов модуля регистрации

Структура БД библиотеки регистрации и идентификации пользователя

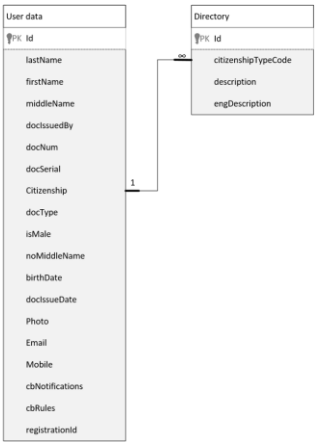


Диаграмма классов

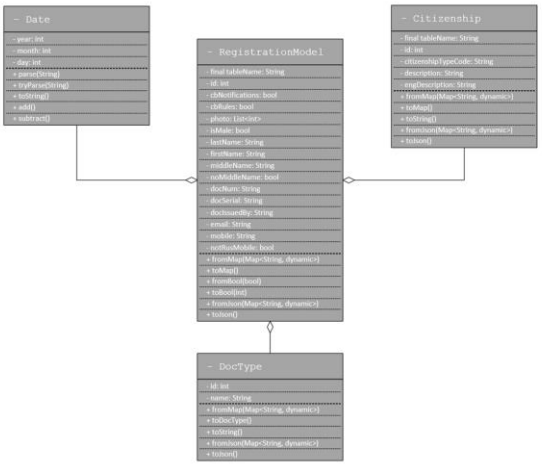


Схема пользовательского интерфейса

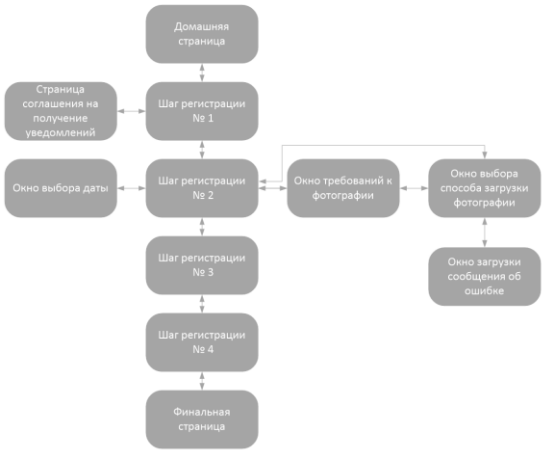
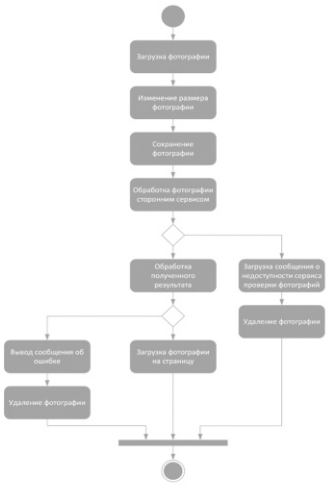
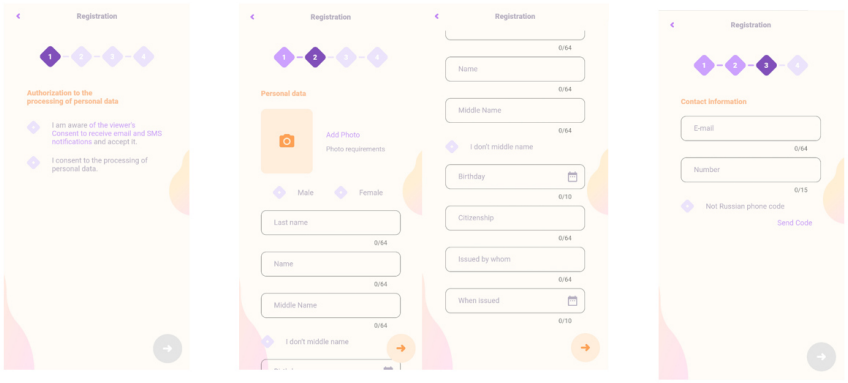


Диаграмма активности при обработки изображения



Работа приложения



Результат работы

1) Проведён анализ современных мобильных операционных систем, а также современных технологий кроссплатформенной разработки мобильного ПО.
2) Разработана структура базы данных библиотеки регистрации и идентификации пользователя для мобильного клиента.
3) Разработан интерфейс библиотеки регистрации и идентификации пользователя.
4) Разработаны асинхронные механизмы обработки, позволяющие осуществлять взаимодействие со сторонним сервисом.
5) Разработано и отлажено мобильное приложение и интерфейс REST API, демонстрирующие работоспособность созданной библиотеки регистрации и идентификации пользователя.

Благодарю за внимание!