**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский

технический университет им. А.Н. Туполева-КАИ»

(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации

Кафедра компьютерных систем

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Образовательная программа: Вычислительные машины, комплексы системы и сети

|  |
| --- |
| **К защите допустить** |
| **Зав. каф. КС** |
| **\_\_\_\_\_\_\_\_\_\_\_\_/Вершинин И.С.** |
| **«\_\_\_» мая 2022 г.** |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему: «Мобильный сервис для выбора профессионального обслуживания дома»

ОБУЧАЮЩИЙСЯ Данилкин А.Д. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(фамилия, имя, отчество) (подпись)*

РУКОВОДИТЕЛЬ ассистент каф.КС, Яковлев И.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(ученая степень, звание, фамилия, имя, отчество) (подпись)*

Mobile service for professional home maintenance

*By*

*Danilkin Alexandr Dmitrievich*

Submitted to the Department of Computer Systems

in partial fulfillment of the Requirements for the degree of

BACHELOR OF SCIENCE

at the

Federal State Budgetary Educational Institution of Higher Education

«Kazan National Research Technical University named after A.N.Tupolev-KAI»

(KNRTU-KAI)

|  |  |  |  |
| --- | --- | --- | --- |
| Author |  |  | Alexandr Dmitrievich Danilkin |
| *(signature)* |  |  |
| Supervisor |  |  | Igor Vladimirovich Yakovlev |
| *(signature)* |  | Assistant, Department of Computer Systems |
| Certified by |  |  | Igor Sergeevich Vershinin |
| *(signature)* |  | PhD, Head of the Department of Computer Systems |
| date |  |  |  |
|  |

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования «Казанский национальный исследовательский технический**

**университет им. А.Н. Туполева-КАИ»**

**(КНИТУ-КАИ)**

Институт (факультет), филиал компьютерных технологий и защиты информации \_\_\_\_\_\_\_

Кафедра компьютерных систем\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление подготовки/специальность\_\_09.03.01 информатика и вычислительная техника

**ЗАДАНИЕ**

на выпускную квалификационную работу

обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

1 Тема выпускной квалификационной работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

утверждена приказом №\_\_\_\_\_\_\_\_\_\_\_ от «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

2 Срок сдачи обучающимся законченной ВКР «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

3 Исходные данные к выпускной квалификационной работе\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4 Перечень подлежащих разработке вопросов и исходные данные к ним:

4.1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5 Перечень графического материала (при наличии):

Презентация на слайдах

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6 Консультанты по ВКР (при их наличии, с указанием относящихся к ним разделов):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование раздела, ФИО консультанта, подпись)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование раздела, ФИО консультанта, подпись)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование раздела, ФИО консультанта, подпись)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование раздела, ФИО консультанта, подпись)*

Дата выдачи задания «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

Руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО)

Задание к исполнению принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО)

**Календарный план выполнения ВКР**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов (разделов)  выпускной квалификационной работы | Срок выполнения  этапов (разделов) ВКР | Примечание |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись) (Фамилия, инициалы) (дата)*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись) (Фамилия, инициалы) (дата)*

Оглавление

[Введение **Ошибка! Закладка не определена.**](#_Toc104328164)

[Глава 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc104328165)

[1.1. Обзор существующих сервисов **Ошибка! Закладка не определена.**](#_Toc104328167)

[1.1.1. Приложение Братья Чистовы 11](#_Toc104328169)

[1.1.2. Приложение Qlean **Ошибка! Закладка не определена.**](#_Toc104328170)

[1.1.3. Приложение Домовенок 13](#_Toc104328175)

[Глава 2. СРАВНЕНИЕ ФРЕЙМВОРКОВ 14](#_Toc104328186)

[2.1. Кросплатформенные фреймворки 14](#_Toc104328187)

[2.1.1. Ionic 15](#_Toc104328189)

[2.1.2. Xamarin 17](#_Toc104328190)

[2.1.3. React Native 19](#_Toc104328191)

[2.1.4. Flutter 20](#_Toc104328192)

[2.2. Разработка пользовательского интерфейса 23](#_Toc104328187)

[2.2.1. Реактивный и декларативный подход 23](#_Toc104328189)

[Глава 3. ВЫБОР ИНСТРУМЕНТОВ РАЗРАБОТКИ 27](#_Toc104328186)

[3.1. Среда разработки Android Studio 27](#_Toc104328187)

[3.1.1. Widget 28](#_Toc104328189)

[3.1.2. Stateless Widget 30](#_Toc104328190)

[3.1.3. Stateful Widget 31](#_Toc104328191)

[3.1.4. Аутентификация Firebase 32](#_Toc104328192)

[3.1.5. Описание языка программирования 33](#_Toc104328192)

[Глава 4. ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ 36](#_Toc104328186)

[4.1. Общее понятие охраны труда 36](#_Toc104328187)

[4.2. Общее положение техники безопасности 38](#_Toc104328187)

[4.3. Техника безопасности программистов 40](#_Toc104328187)

[4.3.1. Рациональное планирование рабочего места программиста 44](#_Toc104328189)

[Глава 5. разработка мобильного приложения 47](#_Toc104328186)

[5.1. Структура приложения 47](#_Toc104328187)

[Заключение 50](#_Toc104328196)

[Список использованных источников и литературы 51](#_Toc104328197)

[Приложение 1 аутентификация в firebase 52](#_Toc104328198)

[Приложение 2 КОД FIREBASE 53](#_Toc104328199)

**АННОТАЦИЯ**

Выпускная квалификационная работа посвящена разработке сервиса для выбора профессионального обслуживания дома с помощью фреймворка Flutter. Рассматривается разработка мобильного приложения на языке Dart, а именно:

* cреда разработки Android Studio;
* архитектура операционной ситсемы IOS и Android;
* работа с пользовательским UI интерфейсом;
* сравнение современных фреймворков;

В заключении представлена экспериментальная часть с результатом в виде программного обеспечения на платформе IOS для выбора обслуживания дома.

**Ключевые слова:** Язык программирования Dart, мобильное приложение, Android и IOS разработка, Firebase, UI интерфейс.

Количество страниц: 50

Количество иллюстраций: 15

Количество таблиц: 1

Количество приложений: 2

Количество использованных источников: 10

**ABSTRACT**

The final qualification work is devoted to the development of a service for choosing professional home maintenance using the Flutter fraimework. The development of useful applications in the Dart language is considered, namely:

* Android Studio development environment;
* IOS and Android system state architecture;
* work with the user interface UI;
* comparison of modern frameworks;

The proposal involves an experimental part with a video program on the IOS platform for home service selection.

**Keywords:** Dart programming language, mobile app, Android and IOS development, Firebase, UI.

Page count: 50

Picture count: 15

Table count: 1

Appendix count: 2

Reference count: 15

**СПИСОК СОКРАЩЕНИЙ**

ОС – Операционная система

ПО – Программное обеспечение

AOT – Ahead-of-Time

API – Application Programming Interface

JSON – JavaScript Object Notation

IDE – Integrated Development Environment

JIT – Just-in-Time

SDK – Software Development Kit

UI – User Interface

# Введение

Мобильные телефоны перестали быть чем-то поразительным и замечательно справляются со своей функцией – являются средством коммуникации между людьми. Смартфоны стали настолько функциональны, что трудно сказать, чего они не умеют. По сути, они стали маленькой копией компьютера, который постоянно можно иметь при себе. В наше время все больше и больше смартфонов и других видов устройств выпускаются на базе OC Android, а так же популярные iPhone на IOS.

Причины распространения этих операционных систем разнообразны:

1. Android поддерживает большое количество устройств разных производителей.
2. OC характеризуется высокой доступностью средств разработки. Средства разработки для платформы Android и IOS бесплатны.
3. Так же преимуществом является наличие бесплатных библиотек для работы со сторонними ресурсами, в то время как для Windows Phone Mobile такие библиотеки не распространены.

Уход за домом отнимает много времени – это большая проблема. Каждый из нас сталкивался с этим в повседневной жизни. Приложение для обслуживания дома поможет это исправить, так как оно позволяет решить ряд проблем.

Раньше, находя разнорабочего по вопросам связанных с обслуживанием дома нужно было искать кого-то кто специализируется на конкретной работе, нужно позвонить кому-то что отнимало наше время.

Сейчас, благодаря развитию технологий с помощью мобильного приложения можно получить доступ к множеству услуг. Спрос на приложения для обслуживания дома быстро растет, так как нас беспокоит повседневный образ жизни, где каждый второй человек обращается за помощью, чтобы управлять своей хозяйственной деятельностью.

В рамках данной работы будет описана разработка мобильного приложения для обслуживания дома OC Android и IOS.

# ГлАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Домашние услуги охватывают широкий спектр промышленных рынков, таких как ремонт и обслуживание, модернизация, уборка дома и другие. Если количество услуг, предлагаемых сектором услуг на дому, растет, масштабы также растут. Услуги по обслуживанию дома включают в себя нематериальные активы, которые предприятия предоставляют для домов. С ростом технологического прогресса возрастает как онлайн так и мобильные платформы бронирования. Рост этой отрасли вызван удобством использования мобильных приложений, что позволяет потребителям заказывать и бронировать услуги одним касанием.

Приложение для оказания услуг ориентирован на помощь в выборе профессиональных мастеров. Приложение предоставляет следующие возможности:

- Услуги легко доступны;

- предоставление услуг в удобное время;

- простая авторизация;

- выбор лучших специалистов.

Для подтверждения конкурентности и необходимости мобильного приложения необходимо провести сравнения с похожими сервисами.

## 1.1. Обзор существующих сервисов

Для сравнения были рассмотрены приложения, которые помогают с выбором сервиса клининг компаний.

Проанализировав данные решения, можно сделать вывод о том, что все программы имеют удобный, отзывчивый и понятный пользовательский интерфейс, возможность посмотреть фото и текстовые материалы, заказать уборку. Не смотря на большое количество полезных функций, имеются и неудобства: отсутствие возможности поиска информации по ключевым словам и датам.

### 1.1.1. Приложение Братья Чистовы

«Братья Чистовы» — это современный клининговый сервис. Его создали люди, которые стремятся к идеальной чистоте и порядку, и любят помогать другим людям.

Основная задача — сделать уборку таким же простым и легким делом, как доставка продуктов и вызов такси. Заказывайте уборку на сайте или через мобильное приложение, отметьте фотографиями важные места в доме и доверьте свой дом профессиональным клинерам.

В «Братьях Чистовых» используют специальные, экологически-чистые моющие средства и профессиональное оборудование (рис. 1.1).



Рис. 1.1. Приложение Братья Чистовы

### 1.1.2. Приложение Qlean

Сервис предоставляет специалистов, которые могут не только убраться в квартире, но и очистить ее от строительной пыли после ремонта и помыть окна. В приложении надо выбрать количество комнат, санузлов, указать адрес и удобное время для выезда клинера. Qlean сам рассчитает стоимость, время уборки и подберет специалиста. Оплата происходит до оказания услуг, но в случае если клиент недоволен результатом — можно обратиться в службу поддержки в течение 24 часов, и уборка будет проведена повторно (риc. 1.2).

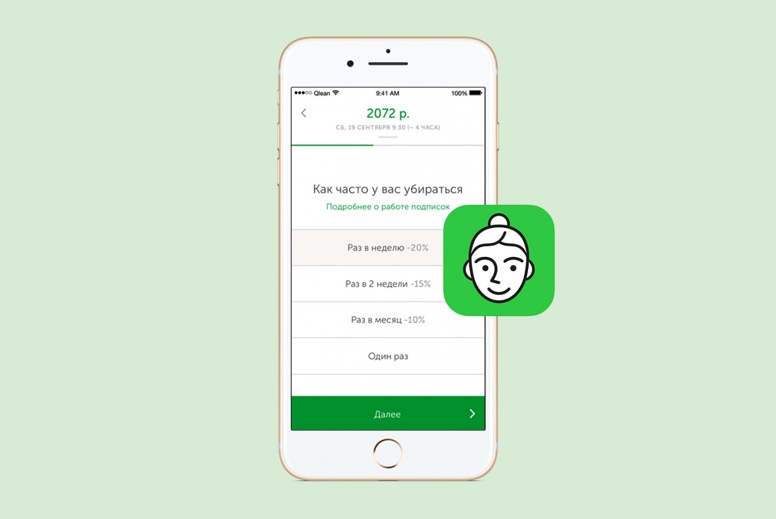


Рис. 1.2 приложение Qlean

### 1.1.3. Приложение Домовенок

Уборка квартир, коттеджей, офисных и производственных помещений. Предлагает широкий перечень услуг: генеральная уборка, поддерживающая, услуги домработницы, услуги мастера, мытье окон и остекление, а также мытье фасадов (рис.1.3).

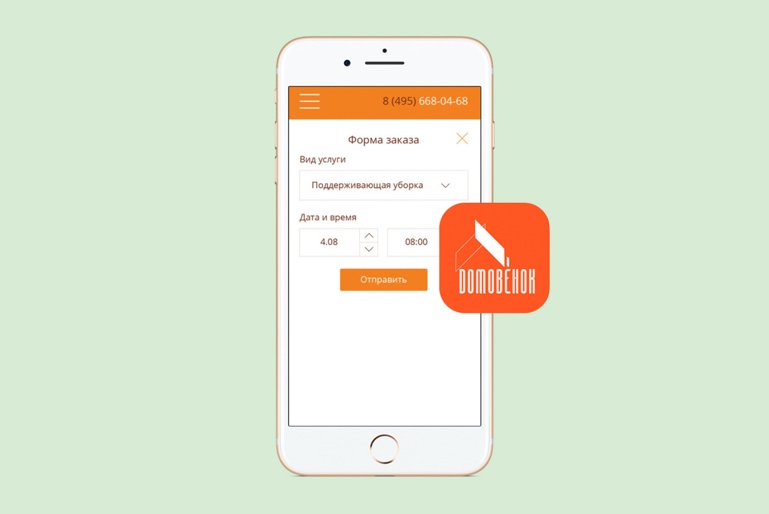


Рис. 1.3. Приложение Домовенок

# ГЛАВА 2. СРАВНЕНИЕ ФРЕЙМВОРКОВ

* + 1. **Кроссплатформенные фреймворки**

В этом разделе рассказывается о четырех основных кроссплатформенных сред разработки, представленных на рынке. Начнем с графика, на котором анализируется тенденция интереса к каждой структуре на основе количества глобальных поисковых запросов в поисковой системе Google за последние пять лет. Диаграмма представлена на (рис. 2.1.)

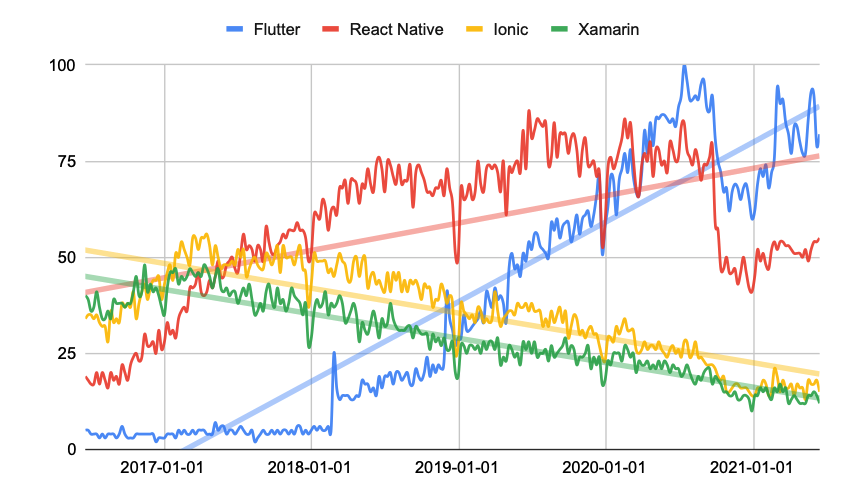


Рис. 2.1. Тенденция мировых исследований за последние 5 лет в отношении четырех основных сред разработки кроссплатформенных приложений.

### 2.1.1. Ionic

Ionic. Это SDK с открытым исходным кодом для разработки гибридных мобильных приложений, основанный на гибридном подходе. Первоначальная версия была выпущена компанией Drifty в 2013 году и построена на основе AngularJS и Apache Cordova. Однако последняя версия была перестроена как набор веб-компонентов, позволяя пользователю выбирать любую структуру пользовательского интерфейса, такую как Angular, React или Vue. Он также позволяет использовать компонентов Ionic вообще без структуры пользовательского интерфейса.

Ionic использует плагины Cordova для получения доступа к функциям операционной системы хоста, таким как камера или GPS. Ionic поддерживает Android, iOS, Windows UWP, веб и рабочий стол - с Electron. Ionic предлагает несколько мобильных компонентов, а также типографику и интерактивные парадигмы. Использование веб-компонентов позволяет Ionic предоставлять настраиваемые компоненты и методы для взаимодействия с ними. Компонент вкладок создает интерфейс с вкладками, который поддерживает навигацию и дизайн в собственном стиле и включает управление состоянием истории. Помимо SDK, Ionic предоставляет такие функции, как развертывание кода и автоматическая сборка, с другими службами, которые могут использовать разработчики, а также с собственной IDE, известной как Ionic Studio.

Ionic поддерживает Android с версии 4.4, а для iOS — с версии 10. Ionic 2 также поддерживает универсальную платформу Windows для создания приложений для Windows 10.

Как видно на рис. 1.1, популярность Ionic в последние годы снижается. Я попытался контекстуализировать это снижение, основываясь на мнениях разработчиков. Есть три наиболее критикуемых аспекта, которые остановили распространение Ionic [1]:

- Слишком много релизов. Разработчики раскритиковали решение Drifty выпустить слишком много версий Ionic и объявить о выпуске Ionic 4, когда Ionic 3 все еще находился в стадии разработки. В результате многие разработчики считали, что разработчики Drifty собирались заново изобрести Ionic с нуля без тщательного планирования.

- Отсутствие документации. Документация Ionic всегда считалась плохой и неполной. Ситуация стала неуправляемой для разработчиков из-за того, что одновременно поддерживалось множество версий Ionic, что делало документацию фрагментированной.

- Проблема производительности. Необходимый для работы Ionic мост Cordova имеет гораздо меньшую производительность, чем нативный код. В случае тяжелых операций ограниченность моста становится еще более очевидной.

### 2.1.2. Xamarin

**Xamarin.** Этот фреймворк изначально был создан разработчиками, создавшими Mono и MonoTouch. Это программное обеспечение стало основой для Xamarin.iOS и Xamarin.Android в 2011 году. Это делает Xamarin старейшей кроссплатформенной средой разработки. В 2013 году была выпущена версия 2.0 SDK с интеграцией с MS Visual Studio. В 2016 году Microsoft приобрела Xamarin и сделала фреймворк открытым. Сегодня Xamarin SDK является частью платформы Microsoft .NET и полностью интегрирован с Visual Studio IDE. Он поддерживает язык программирования C# и XAML в Xamarin.Forms для создания пользовательского интерфейса. Однако тяжелая графика и сложная анимация требуют множества настроек пользовательского интерфейса для каждой платформы. Xamarin можно использовать только с Visual Studio. Это ограничивает компанию в выборе IDE, каждый программист должен научиться пользоваться Visual Studio.

Xamarin упрощает повторное использование бизнес-логики и, таким образом, создает общий скелет для кроссплатформенных приложений. Когда Microsoft купила Xamarin, они намеревались активно продвигать приложения для Windows, поэтому имело смысл оставить разработку пользовательского интерфейса для нативного кода. К сожалению, сегодня этот выбор не позволяет делиться большей частью необходимого кода.

Как видно на рис. 2.1, популярность Xamarin в последние годы снижается в отношении фреймворка Ionic. В последние годы разработчики высказали много критических замечаний, которые можно свести к следующим двум пунктам [2]:

- Путаница с Xamarin. Существует несколько платформ для Xamarin. Раньше были Xamarin.iOS и Xamarin.Android, теперь они завернуты в Xamarin.Native. Затем в 2020 году был выпущен Xamarin.Forms, который продвигал инновационный подход к повторному использованию кода пользовательского интерфейса, но был несовместим с Xamarin.Native и вместо этого полагался на Xamarin.Essential.

Было объявлено, что Microsoft будет называться MAUI — Multiform App User Interface, который будет выпущен в конце 2021 года. Результатом этого выбора стало полное замешательство для разработчиков.

- Трудность и сложность. Многие разработчики комментируют Xamarin, описывая его как излишне сложный и громоздкий. Отладка сложна и часто дает сбои, ведение журнала неполное и тяжелое, для полноценного инструмента требуется несколько IDE — Visual Studio для компиляции, Android Studio для разработки пользовательского интерфейса и JetBrains Rider для кодирования.

### 2.1.3. React Native

Это кроссплатформенный фреймворк с открытым исходным кодом, созданный Facebook для разработки мобильных приложений. React Native родился благодаря опыту, полученному при разработке ReactJS. React Native поддерживает разработку приложений для Android, iOS, MacOS, Windows, Web, UWP, Android TV и TvOS и сочетает в себе язык и навыки ReactJS с возможностями нативной версии.

React Native работает практически идентично фреймворку React, за исключением того, как React Native манипулирует DOM, а не через виртуальный DOM. React Native DOM работает в фоновом режиме, интерпретируя код JavaScript непосредственно на конечном устройстве. Связь с собственной платформой осуществляется через сериализованные данные по асинхронному мосту. Этот мост является основой функций и возможностей React Native.

Нативные компоненты покрывают компонентами React таким образом, что они взаимодействуют с нативными API, используя парадигму декларативного пользовательского интерфейса и JavaScript. Это позволяет разработчикам Javascript разрабатывать нативные приложения и позволяет существующим нативным командам работать намного быстрее. Стиль React Native имеет синтаксис, аналогичный CSS, но не использует HTML или CSS. Вместо этого сообщения JavaScript используются для управления собственными представлениями. Кроме того, React Native позволяет разработчикам писать собственный код внутри проектов React Native, что делает их еще более гибкими.

Как видно на рис. 1.1, популярность React Native в последние годы росла. ReactJS — одна из самых популярных сред веб-разработки, и это сильно повлияло на популярность React Native. Многим разработчикам нравятся React Native и Javascript. Кроме того, Javascript зарекомендовал себя как новый отраслевой стандарт во всем мире, а React Native имеет доступ к крупнейшему в мире сообществу и миллионам библиотек. Однако рост React Native оказался не таким большим, как надеялся Facebook. По сравнению с нативной разработкой, React Native имеет более низкие показатели использования.

На промышленном уровне React Native является крупнейшим конкурентом Flutter. На данный момент это единственная крупная кроссплатформенная структура, которая работает исправно и имеет большие возможности для роста.

### 2.1.4. Flutter

Flutter — это кроссплатформенная платформа пользовательского интерфейса с открытым исходным кодом, разработанная Google и выпущенная в 2018 году. Flutter позволяет разрабатывать приложения для Android, iOS, Web, Windows, MacOS и Linux из единой кодовой базы.

Во время презентации релиза основное внимание Flutter уделялось предоставлению возможности последовательного рендеринга со скоростью 120 кадров в секунду. В 2021 году была выпущена версия 2.0, которая стала первой версией, в которой была добавлена официальная поддержка веб-приложений и настольных приложений для Windows, macOS и Linux, а также улучшена функция добавления в приложение.

Во время разработки приложения Flutter запускаются на виртуальной машине, которая предлагает горячую перезагрузку изменений с отслеживанием состояния без необходимости полной перекомпиляции. Для выпуска приложения Flutter компилируются непосредственно в машинный код, будь то инструкции Intel x64 или ARM, или в JavaScript, если они ориентированы на Web. Фреймворк с открытым исходным кодом, с разрешающей лицензией BSD и имеет множество сторонних пакетов, дополняющих функциональность основной библиотеки.

Успех Flutter был незамедлительным, как показано на рис. 1.1. Сообщество разработчиков приветствовало флаттер за его потенциал.

Flutter разработан как многоуровневая и расширяемая система. Многоуровневая, потому что это система, структурированная таким образом, что различные службы инфраструктуры разделены на различные уровни компонентов, где каждый компонент выполняет определенную четко определенную задачу. Эта конструкция способствует модульности. Вместо этого Flutter является расширяемым, поскольку каждая функция высокого уровня существует в виде серии независимых библиотек, известных как пакеты, где каждая из них зависит от базового ядра, называемого Flutter Engine. По этой причине фреймворк Flutter относительно мал. Разработчики сами выбирают, что добавлять, и в долгосрочной перспективе это помогает сохранить размер приложения небольшим. Flutter состоит из трех основных слоев:

1.Flutter Engine. Он предоставляет примитивы, необходимые для поддержки всех приложений Flutter, и в основном написан на C++. Всякий раз, когда необходимо нарисовать новый кадр, движок отвечает за растеризацию — преобразование изображения из формата векторной графики в растровое изображение — сцены, составленной деревом виджетов. Он обеспечивает низкоуровневую реализацию основного API Flutter, включая графику (через Skia), макет текста, файловый и сетевой ввод-вывод, поддержку специальных возможностей, архитектуру плагинов, а также среду выполнения Dart и цепочку инструментов компиляции. Движок взаимодействует с инфраструктурой Flutter через dart:ui, который заключает базовый код C++ в классы Dart. Эта библиотека предоставляет примитивы самого низкого уровня, такие как классы для управления подсистемами ввода, графики и рендеринга текста.

2. The Flutter Framework. Разработчики взаимодействуют с Flutter через инфраструктуру Flutter, которая представляет собой современную реактивную систему, написанную на языке Dart. Он включает в себя богатый набор платформ, макетов и базовых библиотек, состоящих из ряда слоев. Работая снизу вверх, мы имеем:

• Базовые классы и службы строительных блоков, такие как анимация, рисование и жесты, которые предлагают широко используемые абстракции поверх базовой основы.

• Уровень рендеринга, обеспечивающий абстракцию для работы с компоновкой. Он используется при построении дерева визуализируемых объектов, которые могут динамически изменяться и автоматически обновляться, отражая изменения в макете.

• Слой виджетов, который отображает объект в слое визуализации, имеет соответствующий класс в слое виджетов. Кроме того, слой виджетов позволяет разработчикам определять повторно используемые пользовательские классы. Это уровень, на котором представлена ​​модель react программирования — реактивное программирование — это модель разработки, построенная на асинхронных потоках данных и распространении изменений.

• Библиотеки Material и Cupertino, которые реализуют дизайн Material и iOS, а также их поведение с помощью всеобъемлющего набора готовых компонентов, основанных на примитивах из слоя виджетов.

3. Embedder для конкретной платформы. Для каждой поддерживаемой платформы существует один встроенный модуль, который обеспечивает точку входа для основных функций ядра; Встраивающее устройство работает в координации с базовой операционной системой для использования поверхностей рендеринга и доступа к вводу пользовательских данных, а также для управления циклом обработки сообщений. Программа для встраивания написана на языке, подходящем для платформы: Java и C++ для Android, Objective-C/Objective-C++ для iOS и macOS и C++ для Windows и Linux. Благодаря уровню встраивания код Flutter можно как интегрировать в существующее приложение, так и распространять как отдельное приложение.

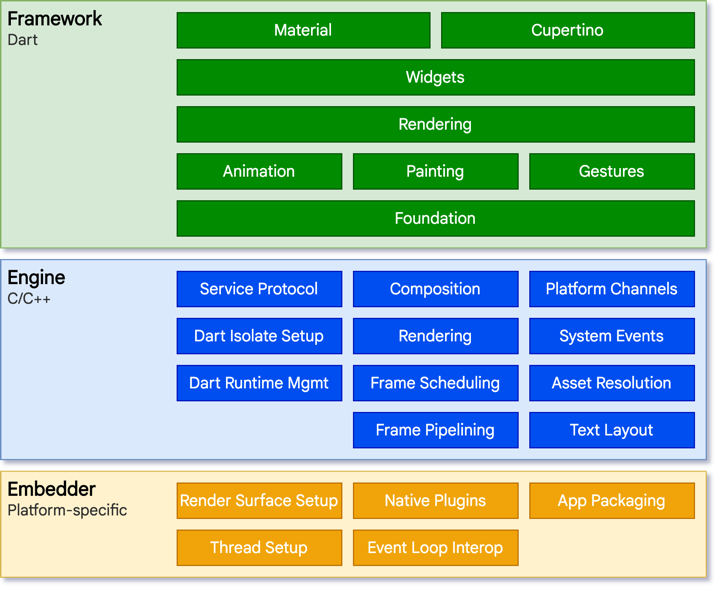


Рисунок 3.1: Графическое представление внутренней архитектуры фреймворка Flutter

Архитектура Flutter представлена графически на рис. 3.1. Здесь можно увидеть три основных слоя, составляющих внутреннюю архитектуру, и их основные обязанности.

Flutter — это кроссплатформенный фреймворк, основанный на скомпилированном подходе, поэтому для работы приложения необходимо включать все слои фреймворка, а это добавляет фиксированные накладные расходы в размере. В частности, согласно описанию в официальной документации, минимальное приложение hello-world, упакованное и сжатое в виде релизного APK, имеет размер примерно 4,3 МБ для ARM и 4,6 МБ для ARM 64. В основном это связано с основным движком, размер которого составляет примерно 3,2 МБ (в сжатом виде), и кодом фреймворка, размером примерно 840 КБ (в сжатом виде). По этой причине плагины платформы, такие как камера или веб-просмотр, доступны в виде пакетов. В конце концов, пакеты также не зависят от платформы, такие как http или пакеты анимации.

**2.2. Разработка пользовательского интерфейса**

**2.2.1. Реактивный и декларативный подход**

Снаружи Flutter — реактивный и декларативный UI-фреймворк. Как было сказано ранее, «реактивный» означает, что внутренняя архитектура Flutter построена вокруг асинхронных потоков данных. Эти потоки необходимы для гибкости в управлении жизненным циклом и поведением компонентов пользовательского интерфейса, а также для распространения состояния. Распространение состояния необходимо для сохранения независимости декларативного пользовательского интерфейса от его внутреннего состояния. Другими словами, Flutter отвечает за обновление интерфейса во время выполнения в результате изменения состояния. С точки зрения разработчика, основная цель состоит в том, чтобы обеспечить сопоставление состояния приложения, предназначенного как данные о поведении приложения, с состоянием интерфейса — данными, представляющими то, как в данный момент отображается пользовательский интерфейс.

Эта модель является довольно новым подходом в мире разработки пользовательского интерфейса и вдохновлена работой Facebook для их собственных фреймворков React и React Native. Эта работа включает в себя серьезное переосмысление многих традиционных принципов проектирования с целью упростить управление интерфейсами, которые обычно очень быстро усложняются с точки зрения графики и поведения.

В большинстве традиционных UI-фреймворков царит императивный подход: начальное состояние пользовательского интерфейса описывается один раз, а затем отдельно обновляется с помощью кода во время выполнения в ответ на события. Параметры в конструкторе компонента используются для описания начального состояния, а затем явно модифицируется поле, относящееся к изменению состояния. Например, виджет A можно инициализировать с помощью нового ключевого слова или получить с помощью метода find\_by\_id(). После этого цвет поля A можно явно изменить, вызвав переназначение поля: instance\_widget\_A.set\_color("новый цветовой код"). Эта система, очевидно, также используется для построения дочернего поддерева виджетов, что создает проблему перемещения параметров вниз по иерархии. Еще одна проблема этого подхода — управление состоянием, которое может стать очень сложным: по мере усложнения приложения разработчикам необходимо отслеживать и знать, как и где изменяется состояние и каковы последствия для всего пользовательского интерфейса. Рассмотрим простое приложение с цветовым полем или ползунком оттенка, используемым для выбора цвета текста: в дополнение к ним рассмотрим также несколько переключателей, чтобы выбрать курсив или полужирный текст, и еще один для графических эффектов, таких как затенение. Есть много мест, где можно изменить состояние, и по мере того, как пользователь взаимодействует с пользовательским интерфейсом, изменения должны отражаться во всех других связанных местах. Незначительные отвлекающие факторы могут создавать запутанные волновые эффекты для несвязанных фрагментов кода, которые трудно предсказать и поддерживать в долгосрочной перспективе. Одним из решений этой проблемы является назначение разных обязанностей различным концептуальным компонентам, как, например, в подходе MVC: пользователь запускает изменения данных в модели через контроллер, а затем модель передает новое состояние в представление через контроллер. Хотя это консолидированный и функциональный подход в контексте клиент-сервер, в рабочем процессе пользовательского интерфейса он невозможен, поскольку в MVC создание и обновление элементов пользовательского интерфейса — это два отдельных шага, которые могут легко рассинхронизироваться: как следствие, непредсказуемое поведение пользовательского интерфейса в может произойти использование в реальном времени.

Решение, принятое Flutter, наряду с другими реактивными фреймворками, использует альтернативный подход, основанный на явном отделении пользовательского интерфейса от его базового состояния. API-интерфейсы Flutter, похожие по стилю на API-интерфейсы React, позволяют разработчикам только описывать, как создавать пользовательский интерфейс. Затем фреймворк использует эту информацию.

- конфигурация виджета - для создания и обновления пользовательского интерфейса по мере необходимости. Виджеты Flutter, опять же похожие на компоненты React, представлены неизменяемыми классами: эти компоненты продвигают чисто функциональный подход и используются для настройки дерева объектов. В иерархическом порядке некоторые виджеты используются для управления специализированным деревом объектов для компоновки, которое становится контейнером для управления другим деревом объектов, предназначенным для композиции — например, свойством цвета текста или соотношением сторон видеоисточника. По сути, Flutter построен на механизмах для эффективного обхода сложных деревьев, преобразования деревьев объектов в объекты более низкого уровня и распространения изменений по этим деревьям. В этом контексте виджет практически является лишь легковесным «чертежом». Виджет, чтобы объявить свой пользовательский интерфейс, должен переопределить свой метод build(), который является функцией, используемой для преобразования его собственного состояния в код пользовательского интерфейса — этот код позже используется для фактического рисования пользовательского интерфейса на холсте Skia. Чтобы внести изменения в пользовательский интерфейс, виджет запускает метод сборки для самого себя — через setState() для StatefulWidgets — и создает новое поддерево виджетов, в котором последнее изменение теперь влияет на пользовательский интерфейс.

Листинг 1.1. Доступное свойство для виджета.

Return Widget{

color: red,

property: valueX,

child: WidgetB(…)

};

Метод build() предназначен для быстрого выполнения и не должен иметь побочных эффектов, вся тяжелая вычислительная работа должна выполняться асинхронно и сохраняться как часть состояния — об управлении состоянием позже. Это позволяет Flutter вызывать его всякий раз, когда это необходимо, возможно, один раз для каждого визуализируемого кадра — аналогично тому, как React обрабатывает DOM и его обновления. В заключение, обновление выполняется путем указания фреймворку заменить виджет в дереве новым обновленным. Следовательно, новые и старые виджеты сравниваются фреймворком для эффективного обновления пользовательского интерфейса. Автоматизированный процесс сравнения Flutter достаточно эффективен, что обеспечивает высокую производительность и способствует интерактивности. Этот подход создает некоторые сложности со стороны памяти, вызванные созданием и удалением множества быстрых объектов, но это решается тем, как язык фреймворка управляет этими сложностями. К счастью, Dart особенно хорошо подходит для этой задачи.

Как объяснялось, во Flutter виджеты являются обязательными для использования подхода декларативной парадигмы: в двух словах, декларативный означает, что мы просто объявляем, как элемент должен отображаться с учетом текущего состояния, фактически не касаясь его. Чтобы сделать это более ясным на практике, лучше думать не о том, как достичь определенного результата, задуманного как поток операторов, которые изменяют состояние программы, а о том, как компонент должен отображать себя в своем новом состоянии.

Структура приложения является наиболее важным фактором для использования декларативного подхода, а декларативный пользовательский интерфейс Flutter полностью построен вокруг виджетов как единицы композиции. Как уже упоминалось, виджет — это легкий «чертеж», базовый строительный блок с широкими возможностями настройки, который представляет собой неизменную декларацию

Композиция получается через иерархическую структуру: от простого текстового виджета вплоть до корневого виджета, каждый из которых вложен внутри своего родителя и получает как контекст, так и ограничения от родителя. Корневой виджет обычно представляет собой контейнер, в котором размещается приложение Flutter, которое обычно представляет собой MaterialApp или CupertinoApp. Оба являются удобными виджетами, которые объединяют ряд виджетов — дизайн и поведение — которые обычно требуются в Google Material Design или Apple Human Interface.

# ГЛАВА 3. ВЫБОР ИНСТРУМЕНТОВ РАЗРАБОТКИ

**3.1. Cреда разработки Android Studio**

Android Studio - это официальная интегрированная среда разработки (IDE) для разработки приложений Android и IOS. Он основан на IntelliJ IDEA, интегрированной среде разработки программного обеспечения на Java, и включает в себя инструменты редактирования кода и средства разработки.

Для поддержки разработки приложений в операционной̆ системе Android Android Studio использует систему сборки на основе Gradle, эмулятор, шаблоны кода и интеграцию с Github.

Каждый проект в Android Studio имеет одну или несколько модальностей с исходным кодом и файлами ресурсов. К этим модальностям относятся модули приложений, модули библиотеки и модули Google AppEngine.

Android Studio использует функцию Instant Push для отправки изменений кода и ресурсов в работающее приложение. Редактор кода помогает разработчику в написании кода и предлагает завершение, рефракцию и анализ кода. Приложения, созданные в Android Studio, затем компилируются в формат APK для отправки в Google Play Store.

Впервые программное обеспечение было анонсировано на Google I/O в мае 2013 года, а первая стабильная сборка была выпущена в декабре 2014 года. Android Studio доступна для настольных платформ Mac, Windows и Linux. Он заменил Eclipse Android Development Tools (ADT) в качестве основной среды разработки приложений для Android.

**3.1.1. Widget**

Каждый пользовательский интерфейс состоит из макета и отдельных компонентов. Макет определяет базовую структуру экрана, например, меню сверху и последующие действия внизу. Затем макет состоит из отдельных компонентов, таких как меню, кнопки или значки. Вместе они создают окончательный интерфейс.

Эти строительные блоки во Flutter называются «виджетами». Будь то простой текст, кнопка или сложные части макета, такие как сетка с несколькими столбцами и строками — все это виджет. Виджеты описывают, как должно выглядеть их представление с учетом их текущей конфигурации и состоянием. Когда состояние виджета изменяется, виджет перестраивает свое описание, которое фреймворк сравнивает с предыдущим описанием, чтобы определить минимальные изменения, необходимые в базовом дереве рендеринга для перехода из одного состояния в другое. Это дерево, называемое «деревом виджетов», на самом деле является одним из трех задействованных деревьев. Фреймворк Flutter использует только один язык для определения как пользовательского интерфейса, так и бизнес-логики.

Widget — это класс Dart, который наследуется от некоторых базовых классов виджета (обычно это StatelessWidget или StatefulWidget). У каждого виджета есть метод build(), который определяет, как виджет должен быть построен (и отрисован на экране).

Виджеты — это не только видимые части пользовательского интерфейса, такие как интерактивные кнопки, текст или значки. Виджеты также определяют макеты, такие как столбцы, строки, сетки, поля между другими виджетами, отступы вокруг них и многое другое как показано на рис. 3.1.

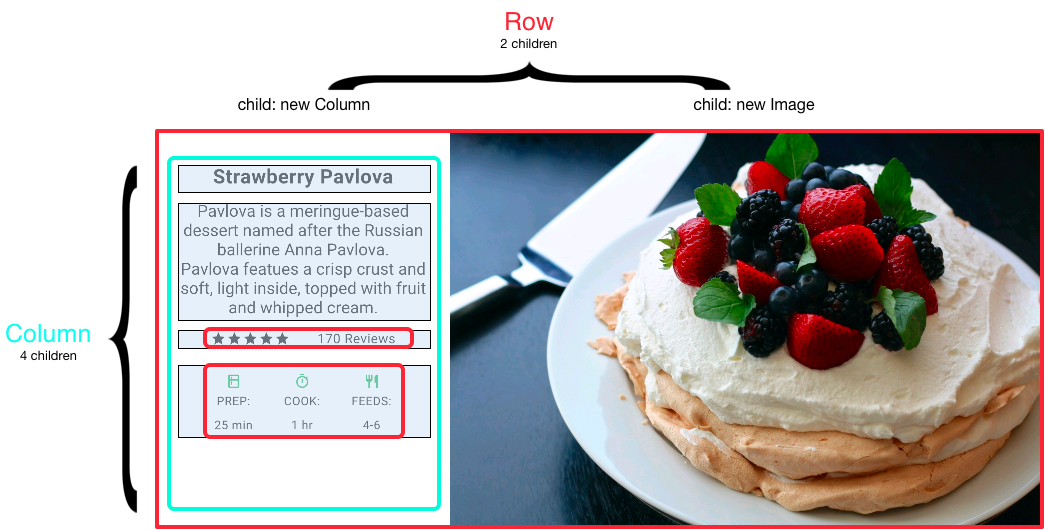


Рис 3.1. Композиция для создания макета

Корневой виджет, виджет Row, содержит два содержательных виджета. Слева это столбец, который содержит больше вложенных виджетов, а справа виджет изображения, который отображает изображение продукта. Разбивку виджета левого столбца можно увидеть на рис. 3.2.

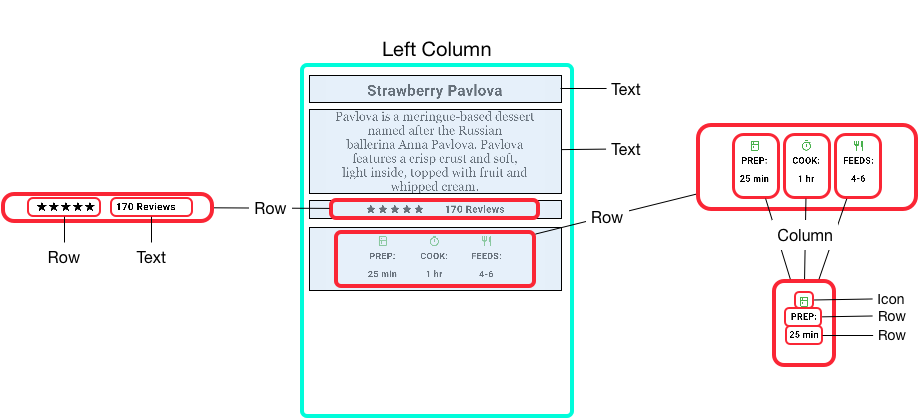


Рис 3.2 Составление виджетов для создания макета

Во Flutter всякий раз, когда изменяется состояние приложения, пользовательский интерфейс перерисовывается. Не существует обязательного способа изменить пользовательский интерфейс, например textWidget.text = 'new text'. Преимущество декларативного подхода заключается в том, что для любого состояния пользовательского интерфейса существует только один путь для кода. Разработчики просто описывают, как должен выглядеть экран в заданном состоянии, и все. Пользовательский интерфейс можно описать как формулу, где пользовательский интерфейс равен функции, которая принимает состояние и возвращает новый пользовательский интерфейс (рис. 3.3).

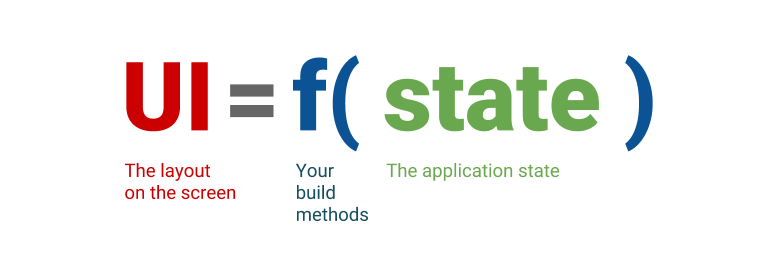


Рис. 3.3. Формула пользовательского интерфейса

**3.1.2. Stateless Widget**

Виджет без состояния - это виджет, который не управляет своим состоянием. Как только он получит свои параметры и будет построен с помощью метода build(), его нельзя будет изменить. Нужно помнить, что всякий раз, когда Flutter решает перерисовать экран, часть дерева перестраивается, но с новыми экземплярами виджетов. Типичными примерами виджета без сохранения состояния могут быть контейнер, текст или значок. Эти виджеты принимают множество параметров, которые могут изменить их внешний вид (и поведение), но сами по себе они не могут быть изменены позже.

Листинг 2.1 Жизненый цикл виджета без сохраненого состояния

class MyWidget extends StatelessWidget {

@override

Widget build(BuildContext context) {

return new Text('my text');

}

}

**StatelessWidget** – нужен там где внутреннее состояние одно и оно сформировано параметрами и данными которые нам известны заранее.

**3.1.3. Stateful Widget**

Всякий раз, когда виджету необходимо управлять своим состоянием и он хочет изменить его, например, в случае события, виджет должен сохранять состояние. Виджет как Stateless принимает параметры, которые можно использовать для настройки этого виджета, но также имеет связанный объект, называемый состоянием. Этот объект состояния является активной частью виджета и используется для изменения виджета и принудительного перестроения пользовательского интерфейса фреймворка. Примером Stateful Widget может быть флажок со статусом «checked».

Stateful Widget имеет не только метод build(), но и связанный с ним объект State, который определяет несколько методов для поддержки жизненного цикла виджета. Этими методами являются initState() для инициализации любого состояния и dispose() для очистки любых выделенных ресурсов.

Объект состояния связан с BuildContext виджета. Эта ассоциация постоянна, и объект состояния никогда ее не изменит. Даже если BuildContext виджета можно перемещать по древовидной структуре, состояние останется связанным с этим контекстом. Это означает, что Stateful Widget может быть заменен во время перестроения дерева новым экземпляром, но объект состояния сохраняется.

Листинг 2.2. Жизненный цикл виджета с отслеживанием состояния

class MyWidget extends StatefulWidget {

@override

createState() => new MyWidgetState();

}

class MyWidgetState extends State<MyWidget> {

@override

initState() {

super.initState();

// ...

}

@override

Widget build(BuildContext context) {

return new Text('my text');

}

}

**3.1.4. Аутентификация Firebase**

Firebase Auth предлагает сервис для защиты приложений со стандартом OAuth 2.0. Он автоматически интегрируется с поставщиками сервисами, такими как Google, Microsoft, Facebook и другими. Кроме того, он предлагает стандартную аутентификацию через электронную почту или телефон пользователя и анонимную аутентификацию, где для каждого пользователя используется случайно сгенерированный идентификатор. Сервис гарантирует, что для каждого пользователя, пока он использует одно и то же устройство, идентификатор остается прежним.

Облачные функции можно использовать для создания Home Service API. Cloud Firestore как база данных (хранилище) и API могут быть защищены с помощью Firebase Authentication. Кроме того, Firebase предлагает услуги по анализу, регистрации и пользовательскому тестированию. Firebase имеет модель «оплата по мере использования» и предлагает бесплатный план (план Spark). План Spark предлагает до 1 ГБ хранилища данных, 20 тысяч операций записи и 50 тысяч операций чтения документов в день. Облачные функции можно вызывать 125 тысяч раз в месяц. Эта цена идеально соответствует требованиям и может свободно использоваться для создания приложения. Если приложение становится популярным и часто используется, фактическая стоимость услуг Firebase рассчитывается на основе текущего использования, и услуги могут быть соответствующим образом увеличены.

**3.1.5. Описание языка программирования**

Dart — это язык программирования, написанный разработчиками Google в 2011 году. Сейчас он достиг второй стабильной версии и постоянно находится в разработке. Dart — это язык программирования общего назначения, одобренный Ecma в качестве стандарта (ECMA-408). Это объектно-ориентированный язык, основанный на классах с синтаксисом в стиле C. Как и другие современные языки высокого уровня, Dart имеет сборщик мусора для освобождения памяти и поддерживает абстрактные классы, интерфейсы, примеси, реифицированные дженерики и вывод типа.

Цель Dart - стать оптимизированным для клиента языком, созданным специально для быстрой разработки приложений на любой платформе. Таким образом, это продуктивный язык программирования для многоплатформенной разработки, который поставляется с гибкой платформой времени выполнения.

Языки определяются их технической оболочкой - выбором, сделанным во время разработки для формирования языка. Dart предназначен для определения приоритетов как разработки на стороне клиента, так и времени разработки. Оптимизация для клиента означает, что Dart делает упор на разработку пользовательских интерфейсов и упрощает ее за счет других аспектов. Например, в Dart нет такого понятия, как RDBMS — Relational Database Management System.

Ключевым аспектом Dart для упрощения разработки пользовательского интерфейса является использование пользовательского интерфейса как кода. При создании пользовательского интерфейса разработчики обычно переключаются между языком разметки пользовательского интерфейса и внутренним языком, на котором они пишут приложение и логику. Обычно это приводит к накладным расходам разработчиков, что снижает производительность. Чтобы уменьшить переключение контекста и развитие фокуса, Dart использует один и тот же синтаксис для определения как чисто логических компонентов - функций, классов, переменных — так и компонентов графического интерфейса. По этой причине у Dart есть некоторые особенности, такие как, например, ключевые слова const и new являются необязательными, что помогает очистить определение виджетов пользовательского интерфейса без использования одного ключевого слова.

Язык Dart имеет строгую статическую типизацию. Во время компиляции он использует проверку статического типа, чтобы всегда проверять, соответствуют ли значения переменных их статическому типу. Этот механизм также известен как звуковая типизация, что означает, что программа никогда не может попасть в состояние, в котором оцениваемое значение выражения не соответствует статическому типу выражения. Однако в Dart аннотации типов необязательны из-за вывода типа. Несмотря на то, что система строго типизирована, Dart позволяет разработчикам использовать динамический тип в сочетании с проверками во время разработки, чтобы ускорить фазу экспериментов.

Начиная с версии 2, выпущенной в 2019 году, Dart обеспечивает нулевую безопасность, что означает, что значение не может быть нулевым, если не указано, что оно может быть. Это работает так же, как система нулевой безопасности Kotlin, а также символ «?» - та же.

Код Dart компилируется DVM - виртуальной машиной Dart. Dart VM работает очень похоже на JVM, но в основном работает в двух режимах. Первый — это режим AOT — компиляция с опережением времени, при котором код Dart компилируется непосредственно в исполняемый код для базовой среды выполнения. В случае мобильного устройства Android и iOS компиляция создаст собственный исполняемый код для платформы ARM, в контексте браузера будет создан код Javascript, который может быть выполнен браузером, а для настольной системы будет создан собственный код. настольный код для платформы X86. В режиме AOT Dart VM не поддерживает такие динамические функции, как загрузка, синтаксический анализ и компиляция исходного кода Dart в режиме реального времени. Он может загружать и выполнять только предварительно скомпилированный машинный код. Виртуальная машина Dart по-прежнему необходима для скомпилированного кода, поскольку она отвечает за предоставление контекст времени выполнения, который включает в себя сборщик мусора и различные собственные методы для работы dart в качестве библиотек, информацию о типах времени выполнения и динамический поиск методов. Режим AOT используется для упаковки и распространения готового приложения.

На этапе разработки код дротика интерпретируется DVM с использованием режима JIT — компиляции Just In Time. DVM может очень быстро компилировать код на низком уровне и может динамически загружать исходный код Dart, анализируя его и компилируя в собственный машинный код на лету для его выполнения. В режиме JIT Dart VM предлагает расширенные функции, такие как горячая перезагрузка, которые упрощают и ускоряют разработку и отладку.

Горячая перезагрузка позволяет разработчикам немедленно запускать последние изменения, внесенные в исходный код, без необходимости перезапуска приложения или потери состояния приложения. Dart VM перезагружает все библиотеки с обновленным кодом, обновляет классы новыми версиями полей и функций, а платформа Flutter автоматически перестраивает, перерисовывает и перестраивает дерево виджетов.

Учитывая опыт, общий для многих разработчиков, у Dart короткая кривая обучения. Типичный опыт мобильного разработчика включает такие языки, как Java, C# и C++ или аналогичные языки, обычно используемые в разработке программного обеспечения. Этого фона достаточно, чтобы начать продвигать Dart как хорошую инвестицию в бизнес, как и другие более известные языки в корпоративном контексте.

**ГЛАВА 4. ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ**

**4.1. Общее понятие охраны труда**

Вопросы охраны труда и техники безопасности регулируются нормативно-правовыми документами Российской Федерации. Хотя каждая сфера производства имеет ряд своих особенностей касательно охраны труда, техники безопасности на рабочем месте и прилегающей территории, существуют общие требования и положения.

К примеру, теоретические основы охраны труда и в принципе само понятие содержится в статье 1 Федерального закона «Об основах охраны труда в РФ» от 17 июля 1999 г. № 181-ФЗ. Данный федеральный закон определяет правовые основы регулирования отношений в области охраны труда между сторонами трудовых отношений, т.е. трудовые отношения работника и работодателя, и нацелен на создание таких условий труда, которые соответствовали бы требованиям сохранения здоровья и жизни работников в процессе трудовой деятельности.

Охрана труда – это система сохранения жизни и здоровья работников в процессе трудовой деятельности, включающая в себя правовые, социально-экономические, организационно-технические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия [3].

Федеральный закон также определяет понятие «условия труда». Условия труда – это совокупностью факторов производственной среды и трудового процесса, оказывающих влияние на работоспособность и здоровье работника [3]. Относительно оказания влияния на работника условия могут быть небезопасными и безопасными. В свою очередь небезопасные условия труда могут быть вредными и опасными. Производственный фактор, воздействие которого на работника может привести к нарушению здоровья и получению заболевания, называется вредным производственным фактором. Если же воздействие производственного фактора может привести к травме, такой производственный фактор является опасным. В случае, если вредного и опасного воздействия на работника в ходе трудового процесса не оказывается или же оно вообще исключено, то такие условия труда являются безопасными. В таком случае на предприятии гарантирована безопасность, т.е. отсутствие недопустимого риска, связанного с нанесением ущерба и вреда здоровью или жизни работника.

Одним из источников воздействия на работника является его рабочее место. Рабочее место – это место, в котором работник находится в связи с его работой и которое контролируется работодателем, будь то прямо или косвенно.

При организации рабочего места в частности программиста на предприятии должны соблюдаться следующие обязательные условия:

- оптимальное размещение оборудования, входящего в состав рабочего места (письменный стол, кресло и персональный компьютер);

- достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;

- наличие естественного и искусственного освещения для выполнения поставленных задач;

- уровень акустического шума не превышает допустимого значения.

Если предприятие добросовестно относится к обеспечению охраны труда, безопасности производственного процесса, не пренебрегает здоровьем и жизнью своих работников, то руководство такого предприятия в ходе проверок получает сертификат соответствия работ по охране труда. Такой сертификат также называется сертификатом безопасности и представляет собой документ, удостоверяющий соответствие проводимых работ по охране труда установленным государственным нормативным требованиям охраны труда.

Каждое предприятие должно организовать работы в сфере обеспечения безопасности трудового процесса таким образом, чтобы управление охраной труда соответствовало главной цели, а именно созданию безопасных условий труда для всего работающего персонала и каждого работника в отдельности.

**4.2. Общие положения техники безопасности**

Понятие «охрана труда» и «техника безопасности» не являются тождественными, т.к. техника безопасности является одной из составных частей охраны труда. Другими словами, техника безопасности – это элемент охраны труда.

Для ознакомления каждого работника с концепцией охраны труда и техники безопасности относительно конкретного предприятия и рабочего места проводятся инструктажи. Всего существует 5 видов инструктажей среди них следующие:

- вводный;

- первичный;

- повторный;

- текущий;

- внеплановый.

Согласно ГОСТ 12.0.004-90 вводный инструктаж проводится со всеми новыми работниками, будь то постоянными или временными, вне зависимости от образования и стажа работы, а также с обучающимися образовательных учреждений, проходящих производственную практику на предприятии. Вводный инструктаж в обязательном порядке проводится только специалистом по охране труда или работником, которого приказом работодателя назначили и возложили эти обязанности. Проведение вводного инструктажа осуществляется в кабинете охраны труда или специально оборудованном помещении с использованием технических средств обучения и наглядных пособий. Продолжительность инструктажа соответствует программе инструктажа и составляет 2 часа. В ходе такого инструктажа главный инженер знакомит работника или практиканта с правилами внутреннего распорядка, основной нормативной документацией, правилами поведения в помещениях и на прилегающей территории. Также освещаются вопросы опасных и вредных производственных факторов в ходе осуществления производственной деятельности конкретного предприятия. Затрагиваются вопросы, касающиеся средств индивидуальной и коллективной защиты. Под такими средствами понимаются технические средства, используемые для предотвращения или сокращения степени воздействия на работников вредных или опасных производственных факторов, а также для защиты от загрязнения. Результаты инструктажа фиксируются в специальном журнале и подтверждаются подписями проводящего и проходящего инструктаж.

Первичный инструктаж обязаны проходить лица, участвующие в производственной деятельности организации. Первичный инструктаж проводится непосредственным руководителем работ на конкретном рабочем месте до начала работником самостоятельной работы. В отличие от вводного инструктажа первичный инструктаж может проводиться в следующих формах:

- индивидуально с каждым работником;

- с группой лиц, обслуживающих однотипное оборудование;

- в пределах общего рабочего места.

Данный инструктаж сопровождается практическим показом безопасных приемов и методов труда. Результаты также оформляются в специальном журнале и подтверждается подписями обеих сторон, - проводящей инструктаж и проходящей инструктаж.

Повторный инструктаж проводится специалистом по охране труда через каждые три месяца. Целью данного инструктажа является обновление знаний касательно безопасного труда и охраны здоровья и жизни. Он проводится по программе первичного инструктажа. И точно также, как и другие виды инструктажа, этот оформляется в журнал и подтверждается подписями.

Текущий инструктаж проводится непосредственно перед началом конкретных работ, на которые требуется допуск. Он также проводится специалистом по охране труда. В отличие от предыдущих видов инструктажа отметка о прохождении этого инструктажа делается непосредственного в наряде на допуск к работам.

Внеплановый инструктаж проводится после нарушений требований охраны труда, повлекших несчастный случай, или при внесении изменении в процесс производства или внедрении новых средств механизации. Отметка о прохождении инструктажа делается главным инженером в специальном журнале, однако, подписи о подтверждении проведения/прохождения не требуются.

После прохождения любого из этих инструктажей проводящий производит устную проверку усвояемости приобретенных знаний и навыков по обеспечению безопасности в процессе производства.

**4.3. Техника безопасности программистов**

Относительно работы в частности техников-программистов можно выделить следующие положения безопасности.

Программист-техник во время работы может быть подвержен воздействию следующих опасных для жизни и здоровья факторов:

- поражение электрическим током;

- травмы.

Персональный компьютер может являться источником ряда вредных и опасных факторов производственной среды. К ним, например, относятся:

- электромагнитные поля, т.е. радиочастоты;

- статическое электричество.

Условия труда при работе на персональном компьютере могут также усугубляться повышенными уровнями шума, неприемлемыми микроклиматическими условиями и недостаточной освещенностью. Работа на персональном компьютере может сопровождаться ограниченной двигательной активностью и монотонией.

При поступлении на работу программист обязан пройти предварительный медицинский осмотр, вводный инструктаж у инженера по охране труда, позднее – первичный инструктаж на рабочем месте, проверку знаний требований безопасности труда в течение 2-14 дней со дня поступления на работу.

Обеспечение безопасности на рабочих местах и всего предприятия в целом зависит от действий каждого работника в отдельности. Поэтому в течение 30 дней со дня поступления на работу работни в обязательном порядке должен пройти обучение и сдать проверку знаний требований охраны труда на предприятии. Затем каждые три года он должен проходить повторную проверку знаний. Повторные инструктаж по охране труда должен проводиться не реже, чем аз в три месяца. Исключение составляют ряд профессии, освобождённых от прохождения инструктажа.

С целью охраны здоровья работников и своевременного выявления отклонений каждый работник обязан проходить периодические медицинские осмотры. Период составляет не реже, чем один раз в два года.

Трудовая деятельности программиста напрямую связана с электроприборами. Такая электро-насыщенность формирует электрическую опасность, источником которой могут быть следующие факторы:

- электрические сети;

- электрифицированное оборудование;

- вычислительная и организационная техника.

В связи с этим риск получения травм, связанных с этим фактором, значительно выше нежели чем у других работников. Электро-травматизм данной сферы производства по сравнению с другими видами производственного травматизма составляет больший процент. Электрический ток, протекая через тело человека, производит термическое, электролитическое, биологическое, механическое и световое воздействие. Различают два вида поражения организма электрическим током: электрические травмы и электрические удары. Поэтому учитывая данные факты, каждый из сотрудников предприятия должен иметь первую квалификационную группу по электробезопасности.

В ходе прохождения регулярных инструктажей работники знакомятся/повторяют требования инструкций по технике безопасности и правилам внутреннего распорядка. Правила внутреннего распорядка в детальных характеристиках зависят от предприятия, однако существуют так называемые общие положения. Например, на абсолютно любом предприятии независимо от направления деятельности или размера штата запрещается употреблять алкогольные, наркотические и/или токсические вещества. Кроме того, запрещается находиться на рабочем месте и прилегающей территории организации даже в нерабочее время в состоянии алкогольного, наркотического или токсического опьянения;

Относительно пожарной безопасности каждый сотрудник обязан уметь пользоваться средствами пожаротушения, а также знать место их нахождения. Однако запрещается использовать пожарный инвентарь не по назначению. В целях избежания пожарной ситуации на территории предприятия запрещается курение табака и разведение открытого огня. Курение табака разрешено в специально предназначенном для этого месте. Такое помещение в обязательном порядке оборудовано ёмкостью с водой и обозначено табличкой «Место для курения».

Касательно оказания медицинской помощи в случае возникновения на то причин. Программист обязан уметь оказывать первую доврачебную помощь и себе, и пострадавшему, используя для этого аптечку и подручные средства.

Что касается непосредственно выполнения работ, программист в праве выполнять только предписанную ему работу, определённую инструкцией по эксплуатации оборудования или должностными инструкциями, утверждёнными генеральным директором, и при условии, что безопасные способы её выполнения хорошо известны. Продолжительность рабочего времени программиста не должна превышать 40 часов в неделю. Продолжительность ежедневной работы определяется правилами внутреннего трудового распорядка, который утверждается генеральным директором.

В случае обнаружения нарушений требований охраны труда другим работником необходимо предупредить его о необходимости их соблюдения. Обнаруженные неисправности оборудования и нарушений требований охраны на рабочем месте необходимо устранять до начала работ. Помимо прочего необходимо немедленно и своевременно извещать ответственного за соблюдение требований охраны труда и безопасности о любой ситуации, угрожающей жизни и здоровью людей, о каждом несчастном случае, происшедшем на производстве, об ухудшении своего здоровья, в том числе о появлении острого профессионального заболевания.

В то же самое время программиста, да и любому другому специалисту, запрещается:

- проходить в местах, не предназначенных для прохода людей;

- заходить без разрешения за ограждения технологического оборудования и опасных зон;

- снимать и перемещать ограждения опасных зон;

- стоять и проходить под поднятым грузом;

- прикасаться к оборванным электропроводам и электропроводам с повреждённой изоляцией;

- проходить под работающими наверху сварщиками, слесарями-ремонтниками, электромонтёрами;

- приступать к выполнению разовых работ, не связанных с его прямыми обязанностями по должности без получения целевого инструктажа.

До осуществления работ программист обязан проверить состояние своего рабочего места. Заострить своё внимание необходимо на следующем:

- правильность подключения оборудования в электросеть;

- освещённость на рабочем месте;

- чистота экрана монитора персонального компьютера. В случае необходимости протереть влажной салфеткой экран;

-правильность установки стола, стула, положения оборудования, угла наклона экрана монитора, положение клавиатуры и, при необходимости, произвести регулировку рабочего стола и кресла, а также расположение элементов компьютера в соответствии с требованиями эргономики и в целях исключения неудобных поз и длительных напряжений тела.

Включать любое электрооборудование разрешается строго только после проведения осмотра. Инженеру запрещается приступать к работе при обнаружении неисправности оборудования. Замеченные недостатки необходимо немедленно устранить. После устранения неисправностей приступить к работе.

Непосредственно вовремя работы программист обязан также выполнять ряд требований техники безопасности. При выполнении производственных заданий за пределами территории предприятия программист обязан соблюдать маршрут, указанный непосредственным руководителем и выполнять только порученную ему работу. При работе на компьютере с текстовой информацией выбирать наиболее физиологичный режим представления черных символов на белом фоне и выбирать оптимальное расстояние от глаз до экрана монитора персонального компьютера, а также следить за освещением.

По окончании работы программисту следует осмотреть рабочее место и привести его в порядок. Оборудование должно быть отключено от сети или переведено в режим, оговоренный инструкцией по эксплуатации. При обнаружении нарушений в ходе проверки пожарной безопасности рабочего места программист обязан сообщить уполномоченному лицу.

В случае невыполнения положений инструкций по охране труда и нарушений техники безопасности, а также неподчинения правилам внутреннего распорядка работник может быть привлечён к ответственности. Например, это могут быт следующие виды мер:

- дисциплинарная ответственность. Работнику могут вынести замечание или выговор, а также уволить с рабочего места;

- материальная ответственность. В этом случае работник обязан возместить нанесённый ущерб предприятию и/или другим работникам;

- уголовная ответственность. Работник подвергается данному виду ответственности в соответствии с законодательством Российской Федерации в зависимости от тяжести последствий.

**4.3.1. Рациональное планирование рабочего места программиста**

Непосредственно рабочее место программиста, как возможный источник оказания вредного и опасного воздействия, также строго регламентируется. Положения нормативов охраны труда распространяются также и на организацию рабочего места программиста и определение оптимальных условий труда программиста.

Рациональное рабочее место обеспечивает программисту удобное положение при работе и высокую производительность труда при наименьшем физическом и психическом напряжении. Под рациональным рабочим местом понимается хорошо приспособленное к трудовой деятельности программиста рабочее место, которое правильно и целесообразно организовано в отношении пространства, формы, размера. Согласно ГОСТ 12.2.032-78 конструкция рабочего места и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. Главными элементами рабочего места программиста являются письменный стол и кресло. Основным рабочим положением является положение сидя. Рабочее место для выполнения работ в положении сидя организуется в соответствии с ГОСТ 12.2.032-78. Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства. Рабочее пространство также ещё называется моторным пространством. Моторное поле – это пространство рабочего места, в котором могут осуществляться двигательные действия человека. Моторное пространство делится на максимальную и оптимальную зоны. Максимальная зона досягаемости рук – это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе. Оптимальная зона – это часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом. Эти зоны, а также рациональное размещение оборудования и документации наглядно представлено на рисунке 4.1.

На данном рисунке горизонтальная плоскость рабочего пространства разделена на зоны согласно двигательной активности работника. Под этими зонами подразумевается следующее:

- зона «а» – это зона максимальной досягаемости;

- зона «б» – это зона досягаемости пальцев при вытянутой руке;

- зона «в» – это зона легкой досягаемости ладони;

- зона «г» – это оптимальное пространство для грубой ручной работы;

- зона «д» – это оптимальное пространство для тонкой ручной работы.

В таблице 1 представлено возможное оптимальное размещение предметов труда и документации в пределах досягаемости рук согласно зонированию двигательной активности.

При проектировании письменного стола необходимо учитывать следующее:

- высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники;

- нижняя часть стола должна быть сконструирована так, чтобы программист мог удобно сидеть, не был вынужден поджимать ноги;

- поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения программиста;

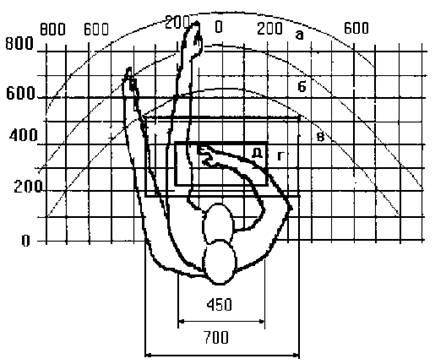


Рис. 4.1. Рациональное планирование рабочего места программиста

Их количество должно составлять не менее трех для хранения документации, листингов, канцелярских принадлежностей, личных вещей.

Таблица 1. Размещение предметов

|  |  |
| --- | --- |
| Предмет труда | Оптимальная зона |
| Дисплей | Зона «а» в центре |
| Клавиатура | Зона «г» или «д» |
| Системный блок | Зона «б» слева |
| Принтер | Зона «а» справа |
| Документация, часто используемая в работе | Зона «в» слева |
| Документация, редко используемая в работе | Выдвижные ящики стола |

Параметры рабочего места выбираются в соответствии с антропометрическими характеристиками. При использовании этих данных в расчетах следует исходить из максимальных антропометрических характеристик.

При работе в положении сидя рекомендуются следующие параметры рабочего пространства:

- ширина не менее 700 мм;

- глубина не менее 400 мм;

- высота рабочей поверхности стола над полом 700-750 мм.

Оптимальными размерами стола являются:

- высота 710 мм;

- длина стола 1300 мм;

- ширина стола 650 мм.

Поверхность для письма должна иметь не менее 40 мм в глубину и не менее 600 мм в ширину.

Важным элементом рабочего места программиста является кресло. При проектировании кресла исходят из того, что при любом рабочем положении программиста его поза должна быть физиологически правильно обоснованной, т.е. положение частей тела должно быть оптимальным.

Важным моментом является также рациональное размещение на рабочем месте документации, канцелярских принадлежностей, что должно обеспечить работающему удобную рабочую позу, наиболее экономичные движения и минимальные траектории перемещения работающего и предмета труда на данном рабочем месте.

Таким образом, охраной труда можно назвать науку о сохранении здоровья и безопасности человека в процессе осуществления трудовой деятельности. Основными методами обеспечения охраны труда являются:

- выявление и идентификация опасных и вредных факторов;

- разработка методов и средств защиты человека путем снижения опасных и вредных факторов до приемлемых значений;

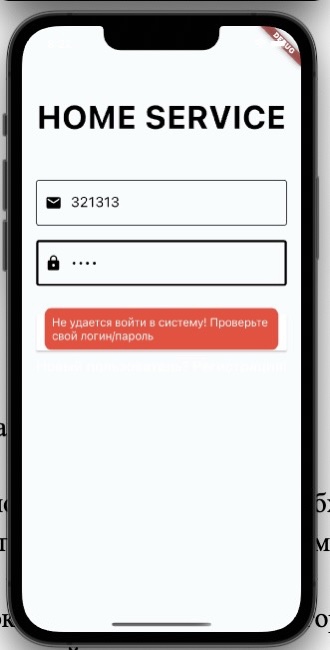
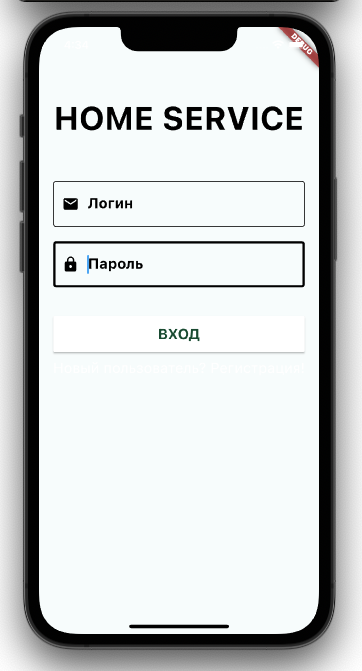
- выработка мер по ликвидации последствий чрезвычайных ситуаций.

**ГЛАВА 5. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ**

* + 1. **Структура приложения**

Первое, что видит пользователь при запуске приложения страницу с входом. Имеется два варианта: вход для тех у кого уже есть учетная запись, и регистрация. Войти в приложение очень просто, нужно указать адрес электронной почты вместе с паролем. Если пароль окажется неверным то это отобразится Рис. 5.1.

Рис 5.1. Окно входа



После проверки логина и пароля и регистрации, у нас открывается превью приложения (с описанием услуг)

Превью представлено на рисунке 5.2.

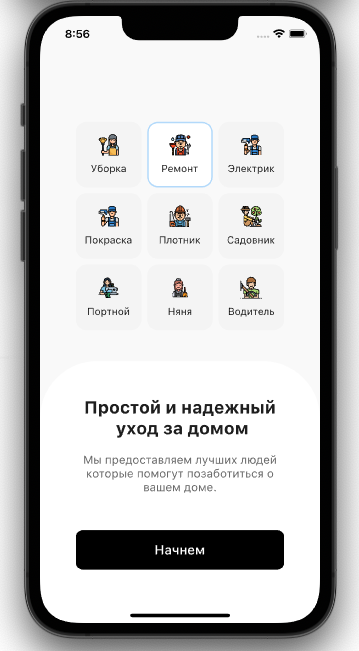
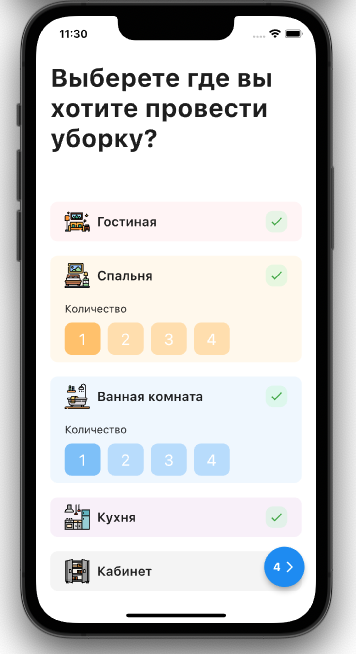
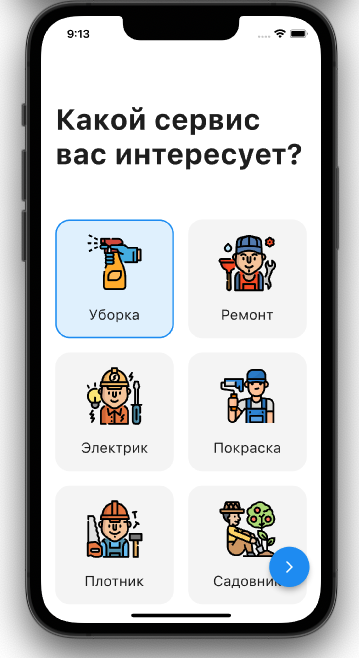


Рис 5.2. Превью

После нажатия на кнопку у нас открывается следующая страница с выбором какой сервис нас интерисует, нажимаем на нее и открывается следующее окно с выбором в каких комнатах необходимо выполнить уборку. Рис.5.3.



а) Рис 5.3.Выбор сервиса б) Рис 5.4. Выбор комнат

После того как выбрали где нужно проводить уборку(рис.5.4.), на странице выбираем удобную дату и время, нужно ли повторить данную услгу и дополнительный сервис.

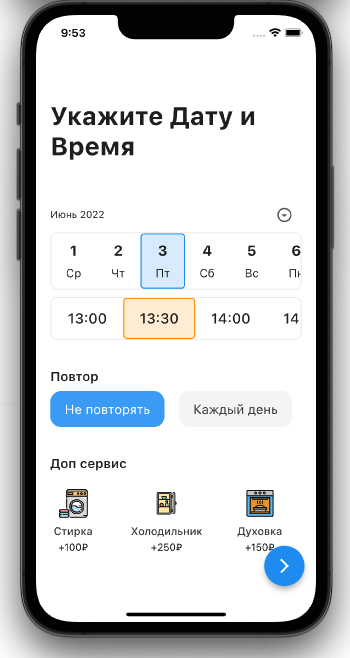


Рис 5.5. Выбор Даты и Времени

После того как выбрали дату и время нажимаем и открывается страница формирования заказа показано на рис 5.6.

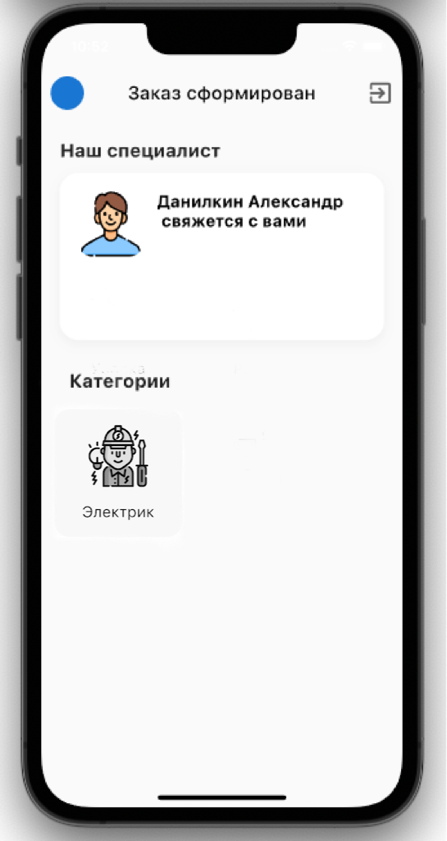


Рис 5.6. Формирование заказа

**ЗАКЛЮЧЕНИЕ**

Целью данной дипломной работы было проектирование и разработка мобильного сервиса для выбора профессионального обслуживания дома.

Итогом дипломной работы является клиентское приложение, позволяющие осуществлять следующие действия:

- предоставление услуг в удобное время;

- простая авторизация;

- выбор лучших специалистов.

Для подтверждения конкурентности и необходимости мобильного приложения было проведено сравнения с похожими сервисами.

Разработанное приложение позволяет облегчить мастерам сбор и формальных, и неформальных данных о клиенте. Благодаря этому улучшается качество обслуживания каждого отдельного клиента, что благотворно влияет на репутацию компании.

В результате работы над выпускной квалификационной работой была изучена среда разработки Flutter. Основное преимущество данной среды заключается в возможности разработки кросс-платформенных приложений в пределах одной среды и одного программного кода.

**CONCLUSION**

The purpose of this thesis was to design and develop a mobile service for choosing a professional service at home.

The result of the thesis is a client application that allows you to perform the following actions:

- provision of services at a convenient time;

- simple authorization;

- selection of the best specialists.

To confirm the competitiveness and the need for a mobile application, comparisons were made with similar services.

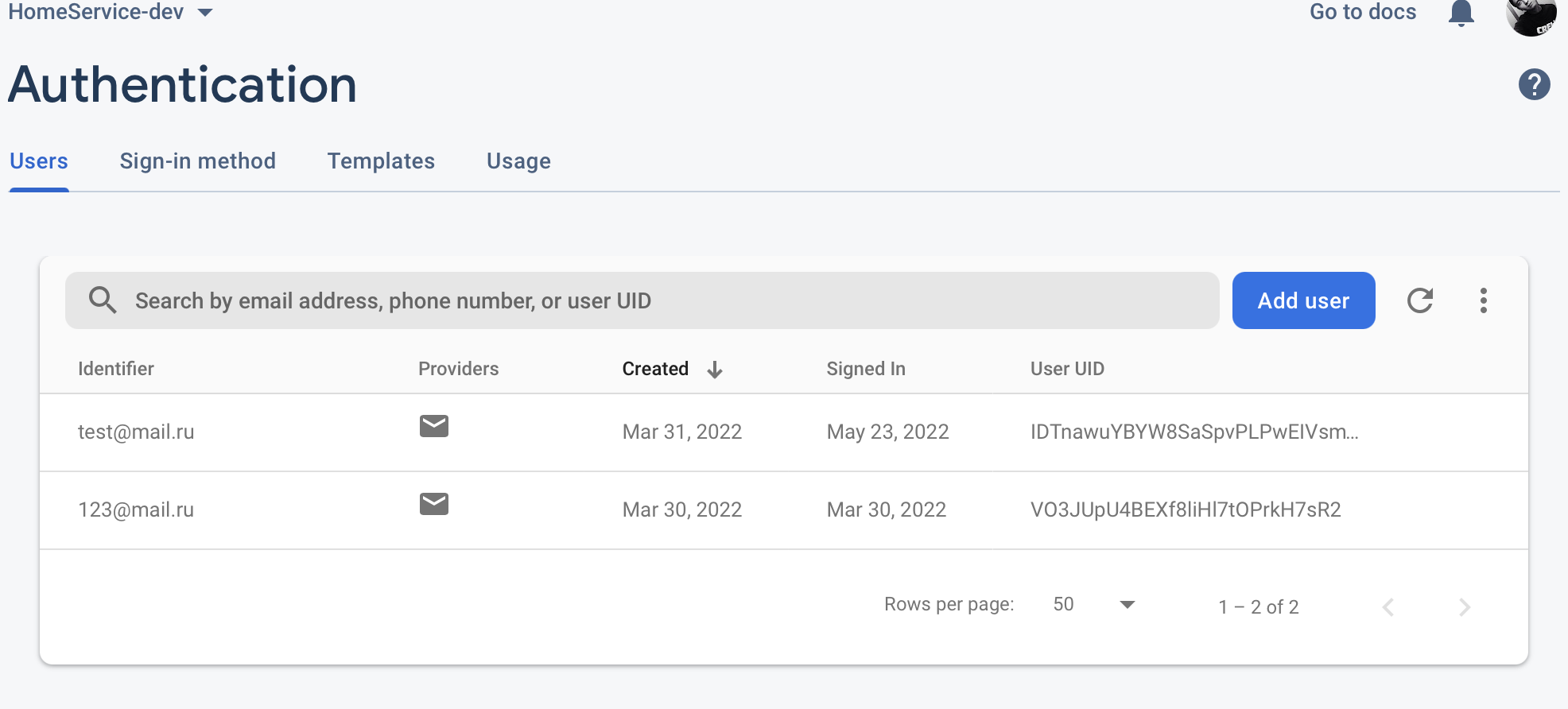
The developed application makes it easier for the masters to collect both formal and informal data about the client. This improves the quality of service for each individual client, which has a beneficial effect on the company's reputation.

As a result of work on the final qualification work, the Flutter development environment was studied. The main advantage of this environment is the ability to develop cross-platform applications within one environment and one program code

# Список использованных источников и литературы

|  |  |
| --- | --- |
| 1. | Dave Francis. Is Driftys lonic dead? [Электронный ресурс] [2021] URL: https://foresightmobile.com/blog/ionic. (дата обращения; 20.04.2022) |
| 2. | Dave Francis. Everything that is wrong with Xamarin [Электронный ресурс] [2021] URL: https://foresightmobile.com/blog/2020/09/15/isxamarindead  .(дата обращения; 25.04.2022) |
| 3. | Федеральный закон от 28 декабря 2013 г. № 426-ФЗ "О специальной оценке условий труда"; |
| 4. | Федеральный закон от 27 июля 2006 г. (ред. от 28 декабря 2013 г.) № 149-ФЗ "Об информации, информационных технологиях и о защите информации"; |
| 5. | Федеральный закон от 17 июля 1999 г. (с изм. от 20 мая 2002 г., 10 января 2003 г., 9 мая, 26 декабря 2005 г.) № 181-ФЗ «Об основах охраны труда в Российской Федерации»; |
| 6. | СанПиН 2.2.2/2.4.1340-03. Санитарно-эпидемиологические правила и нормативы «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»; |
| 7. | Android Development Tool. [Электронный ресурс] URL: https://developer.android.com (дата обращения: 30.04.2022) |
| 8. | Dart language overview [Электронный ресурс] URL: https://dart.dev/guides/language (дата обращения: .04.2022) |
| 9. | Об опыте разработки на Dart [Электронный ресурс]. / Evrone // VC. - 2019- URL: https://vc.rn/evrone/57589-dart (дата обращения 20.03.2022) |
| 10. | Среда разработки Android Studio [Электронный ресус] URL: https://androidstudio.ru/ (дата обращения 18.03.2022) |

# Приложение 1 аутентификация в firebase



# Приложение 2 Код FIREBASE

class AuthorizationPage extends StatefulWidget {  
AuthorizationPage({Key? key}) : super(key: key);  
  
[@override](https://vk.com/id483219527)  
\_AuthorizationPageState createState() => \_AuthorizationPageState();  
}  
class \_AuthorizationPageState extends State<AuthorizationPage> {  
TextEditingController \_emailController = TextEditingController();  
TextEditingController \_passwordController = TextEditingController();  
  
late String \_email;  
late String \_password;  
bool showLogin = true;  
  
AuthService \_authService = AuthService();  
  
[@override](https://vk.com/id483219527) //заголовок  
Widget build(BuildContext context) {  
Widget \_logo() {  
//добавляем лого  
return Padding(  
padding: EdgeInsets.only(top: 100), //отступы только сверху  
child: Container(  
//основной метод  
child: Align(  
//центрированное лого (по умолчанию)  
child: Text('HOME SERVICE',  
style: TextStyle(  
fontSize: 45, //размер  
fontWeight: FontWeight.bold, //жирный  
color: [Colors.black](https://vk.com/away.php?utf=1&to=http%3A%2F%2FColors.black" \t "_blank))) //цвет  
)));  
}  
Widget \_input(Icon icon, String hint, TextEditingController controller,  
bool obscure) {  
return Container(  
padding: EdgeInsets.only(left: 20, right: 20),  
child: TextField(  
controller: controller,  
obscureText: obscure, //контроль пароля  
style: TextStyle(fontSize: 20, color: [Colors.black](https://vk.com/away.php?utf=1&to=http%3A%2F%2FColors.black" \t "_blank)),  
decoration: InputDecoration(  
hintStyle: TextStyle(  
fontWeight: FontWeight.bold,  
fontSize: 20,  
color: [Colors.black](https://vk.com/away.php?utf=1&to=http%3A%2F%2FColors.black" \t "_blank)),  
hintText: hint,  
focusedBorder: OutlineInputBorder(  
borderSide: BorderSide(color: [Colors.black](https://vk.com/away.php?utf=1&to=http%3A%2F%2FColors.black" \t "_blank), width: 3)),  
enabledBorder: OutlineInputBorder(  
borderSide: BorderSide(color: [Colors.black](https://vk.com/away.php?utf=1&to=http%3A%2F%2FColors.black" \t "_blank), width: 1)),  
prefixIcon: Padding(  
padding: EdgeInsets.only(left: 10, right: 10),  
child: IconTheme(  
data: IconThemeData(color: [Colors.black](https://vk.com/away.php?utf=1&to=http%3A%2F%2FColors.black" \t "_blank)), child: icon))),  
),  
);