# **Техническое задание**

Название проекта: Информационная система «Ремонт средств связи»

Целью данного проекта является создание мобильного приложения для организации, осуществляющей ремонт средств связи (СС). Приложение будет разработано с использованием фреймворка Flutter.

Мобильное приложение позволит организации управлять следующими аспектами процесса ремонта:

- Авторизация пользователя

- Запись на аренду техники в ремонт

- Создание квитанций для клиентов

- Запись информации о процессе ремонта

Разрабатываемое приложение должно работать на самых популярных на сегодняшний день мобильных операционных систем – Android и Ios с минимальной версией 5.0 и 12.0 соответственно.

Языком разработки данного приложения является Flutter — это популярный язык разработки мобильных приложений, созданный Google и первоначально выпущенный в 2017 году. Это платформа с открытым исходным кодом, которая позволяет разработчикам разрабатывать высокопроизводительные кроссплатформенные мобильные приложения для устройств iOS и Android. Язык основан на языке программирования Dart, который прост в изучении и понимании. Flutter приобрел популярность благодаря своей функции горячей перезагрузки, позволяющей разработчикам вносить изменения в код в режиме реального времени. Это экономит время и повышает производительность. Flutter также предоставляет богатую коллекцию виджетов, позволяющую разработчикам создавать красивые и отзывчивые пользовательские интерфейсы. В связи с растущим спросом на разработку мобильных приложений Flutter — отличный выбор для разработчиков, которые хотят быстро и эффективно создавать высококачественные приложения.

Мобильное приложение должно иметь простой и понятный пользователю интерфейс.

* Для разработки такого мобильного приложения необходимо:
* операционная система MacOs 12(Monterey);
* процессор: 3,0 ГГц 3,0 ГГц, 6‑ядерный Intel Core i5 и выше;
* оперативная память: 16 Гб и больше;
* пропускная способность сетевого интерфейса: 1Гбит/c;
* свободное дисковое пространство: 128 Гб и больше.
* Также необходимо следующее программное обеспечение:
* текстовый редактор Visual Studio Code;
* интегрированная среда разработки Android Studio (Bumblebee и выше);
* интегрированная среда разработки Xcode (14.2 и выше);
* IOS Simulator (IOS 12 и выше);
* Android SDK (21 и выше);
* Разработка должна быть проведена в несколько стадий:
* технической задание;
* технический (и рабочий) проект;
* внедрение (выгрузка приложения в магазин приложений).

Выходные данные в мобильном приложении представлены в виде базы данных клиентов, подавших заявку на ремонт. Она состоит из следующих пунктов:

* ФИО клиента;
* номер телефона клиента;
* адрес клиента;
* номер паспорта клиента.

Эти данные пользователь самостоятельно заносит и подает заявку.

Выходные данные должны быть представлены в виде вывода графической и информации на экран мобильного устройства в результате анализа программного обеспечения. К выходным данным относятся все данные, занесенные пользователем в базу данных.

**Рефакторинг Кода**

В современном мире существует большое множество языков программирования, которые позволяют создавать новые и усовершенствовать, ранее созданные, программы и системы.

Рефакторинг кода — это процесс улучшения структуры, удобочитаемости и удобства сопровождения существующего кода. Это важная практика для повышения качества кода, уменьшения технического долга и увеличения скорости разработки. Рефакторинг может быть выполнен в коде любого приложения, включая мобильное приложение.

Рефакторинг выполняет при следующих проблемах:

* «мертвый код». Переменная, параметр, метод или класс больше не используются: требования к программе изменились, но код не почистили. Мёртвый код может встретиться и в сложной условной конструкции, где какая-то ветка никогда не исполняется из-за ошибки или изменения требований. Такие элементы или участки текста нужно очистить;
* дублирование. Один и тот же код выполняет одно и то же действие в нескольких местах программы. Здесь требуется вынести эту часть в отдельную функцию;
* имена переменных, функций или классов не передают их назначение. Имена должны сообщать, почему элемент кода существует, что он делает и как используется.
* слишком длинные функции и методы. Если получается больше 10-20 строк кода, требуется разделить функцию на несколько маленьких и добавить одну общую. Здесь маленькие функции выполняют по одной операции, а общая функция их вызывает;
* слишком длинные классы. То же самое. Оптимальная длина класса – 20–30 строк. Следует разбить длинный класс на несколько маленьких и включить их объекты в один общий̆ класс;
* слишком длинный список параметров функции или метода. Если все эти параметры действительно нужны, нужно вынести их в отдельную структуру или класс с понятным именем, а в функцию передать ссылку на него;
* много комментариев. При желании пояснить какой-то участок кода, стоит попробовать сначала его переписать, чтобы и так стал понятным. Бесполезные комментарии загромождают программу.

Рефакторинг стоит выполнять постоянно и немного.

Рефакторинг кода мобильного приложения во Flutter является важным процессом для улучшения качества кода, сокращения технического долга и

увеличения скорости разработки. Он включает в себя анализ существующего кода, определение областей улучшения, планирование процесса рефакторинга, рефакторинг кода, тестирование рефакторингового кода и развертывание рефакторингового кода. Выполнение этих шагов гарантирует высокое качество приложения, хорошую производительность и простоту обслуживания.

В мобильном приложении «Ремонт средств связи» рефакторинг кода происходит в навигации. Для навигации в языке разработки Flutter есть стандартный пакет. Но он не может похвастаться удобством и чистотой по сравнению с «auto\_route».

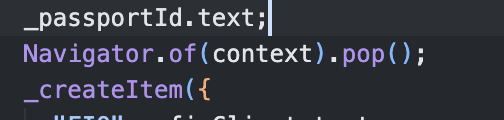
С помощью «auto\_route» разработчики могут определять маршруты с помощью аннотаций и автоматически генерировать навигационный код. Это устраняет необходимость ручного кодирования маршрутов и снижает вероятность ошибок.

Таким образом, стандартная навигация была заменена на навигацию с пакетами «auto\_route», «auto\_route\_generator» и «build\_runner» (Снимок 1, Снимок 2, Снимок 3, Снимок 4).

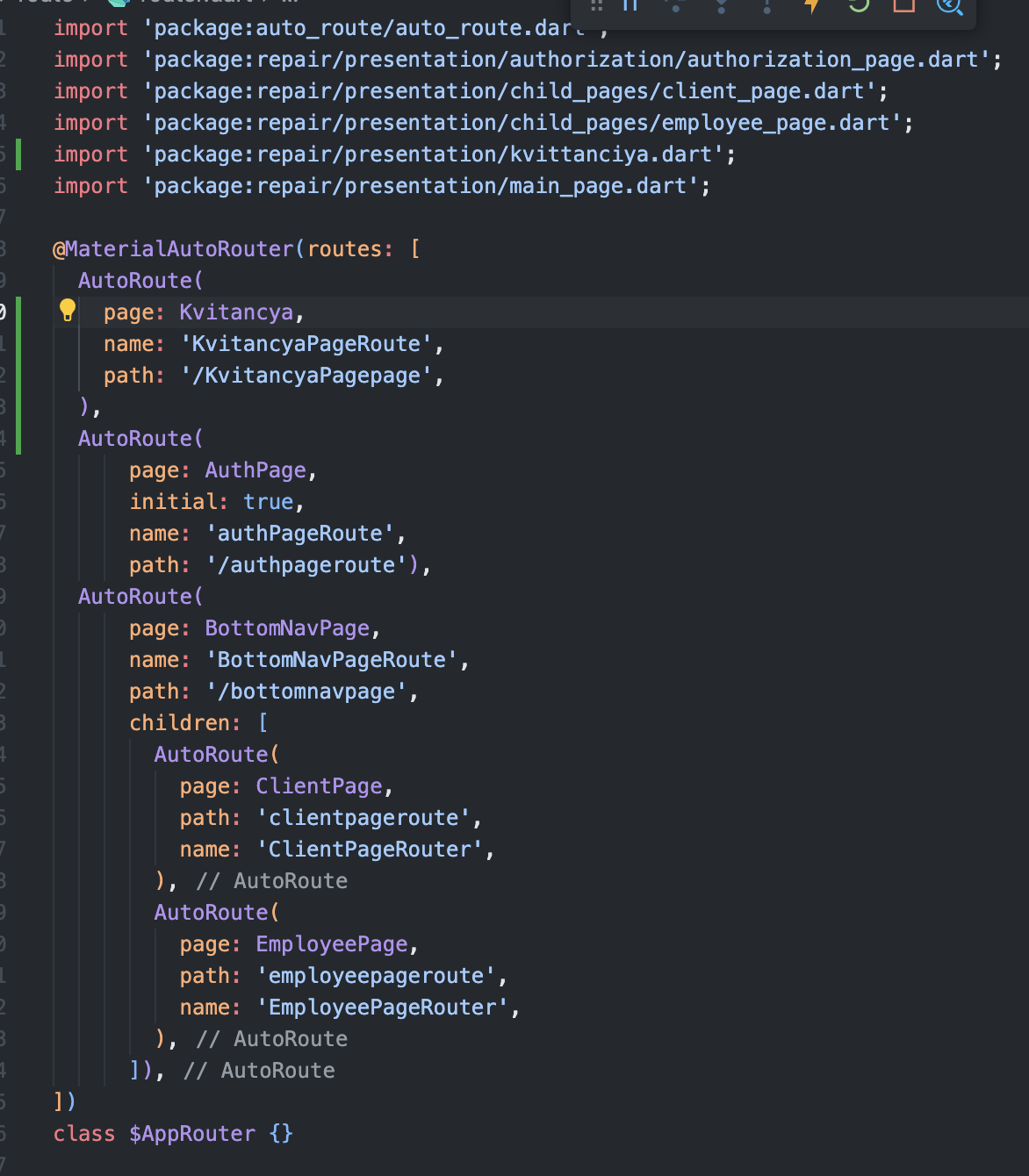


Снимок 1 – стандартная навигация Flutter.

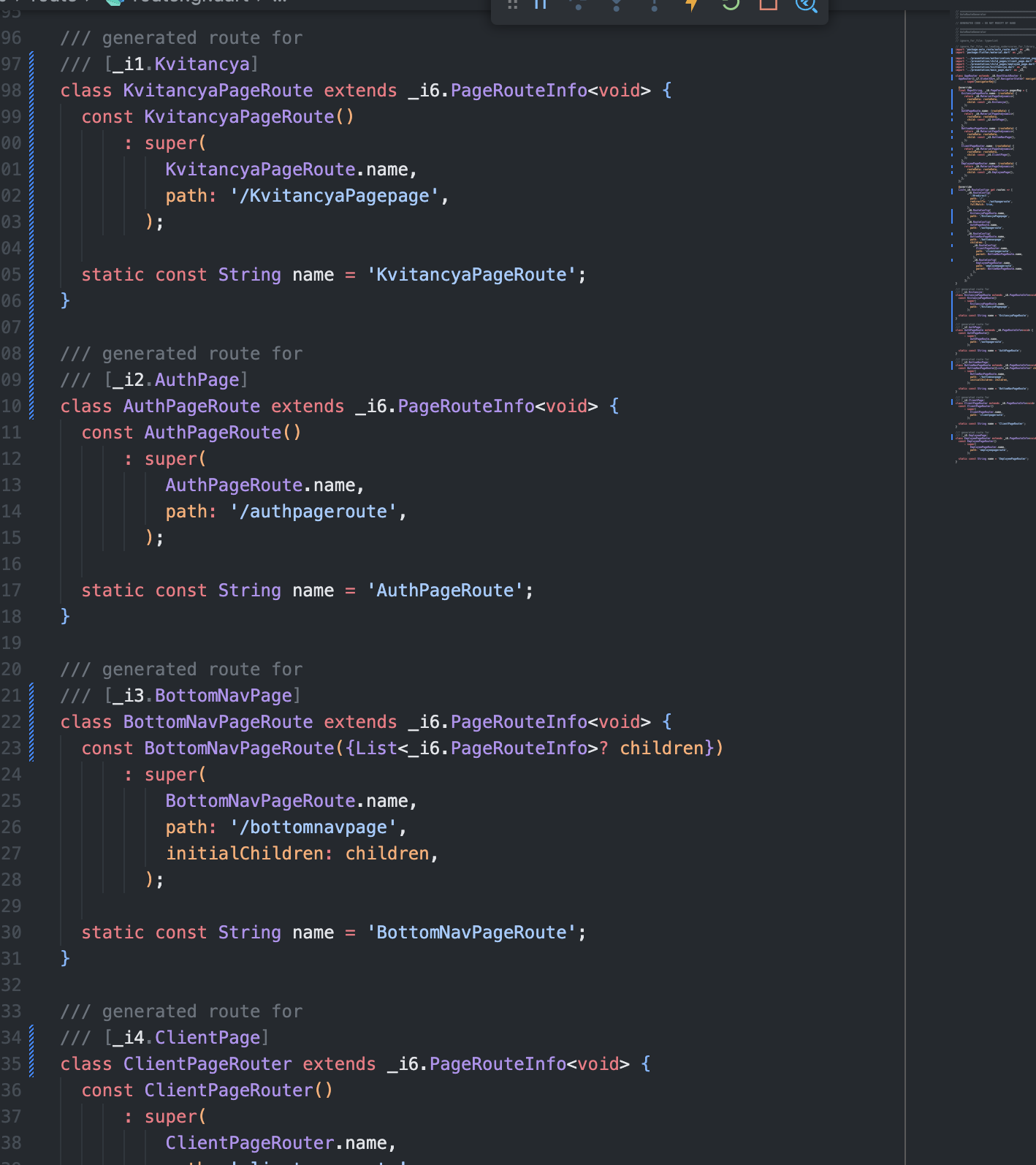
Стандартная навигация языка Flutter не имеет «вложенной навигации», которая необходима для приложения «Travel Zone».



Снимок 2 – навигация во Flutter с пакетами «auto\_route», «auto\_route\_generator», «build\_runner».



Снимок 3 – построение навигации с помощью пакета «auto\_route».



Снимок 4 – генерация маршрутов навигации пакетами «auto\_route\_generator» и «build\_runner».

Таким образом, поменяв стандартную навигацию на возможности пакета «auto\_route» мы получаем вложенную навигацию с большими возможностями, например, анимация и задержка перехода.

**Тестирование программного кода**

Тестирование на вход приложение:



Рисунок 5 – тестирование приложения.