

**МИНИСТЕРСТВО ЦИФРОВЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ
УЗБЕКИСТАН
ФЕРГАНСКИЙ ФИЛИАЛ
ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛ-ХОРАЗМИЙ**

**ФАКУЛЬТЕТ «ПРОГРАММНАЯ ИНЖЕНЕРИЯ И ЦИФРОВАЯ
ЭКОНОМИКА»
КАФЕДРА «ПРОГРАММНЫЙ ИНЖИНИРИНГ»**

«УТВЕРЖДАЮ»

Зав. Кафедрой в.и.:

_____ Х.Ш.Мусаев

«_____» _____ 2024 г.

КВАЛИФИКАЦИОННАЯ ВЫПУСКНАЯ РАБОТА

на тему:

«Создание мобильное приложение управление торговли в магазинах»

Выполнил:

Ботиралиев Б.Б.

(подпись)

Студент группы 655 - 20 ПИ

Руководитель:

Хусанов.Б

(подпись)

Фергана – 2024

АННОТАЦИЯ

Данную выпускную квалификационную работу выполнил студент группы 655-20 Ботиралиев Бахтиёр. В данной ВКР, используя для серверной части высокоуровневый язык программирования Python и библиотеку FAST API для сервиса автоматизации управления торговлей в магазинах, а также язык Dart и его фреймворк для кроссплатформенной разработки, такой как Flutter, разработано мобильное приложение. Flutter самособой кроссплатформенный фреймворк который компилирует на все платформы . Во введении определены цели и задачи исследования. В теоретической части анализируются основные концепции автоматизации управления торговлей. В проектной части представлено описание разработки и архитектуры приложения. Заключение подводит итоги и предлагает направления для дальнейших исследований.

ANNOTATION

This final qualifying work was completed by a student of group 655-20 Botiraliiev Bakhtiyor. In this work, using the high-level Python programming language for the server part and the FAST API library for the automation service for managing trade in stores, as well as the Dart language and its framework for cross-platform developments such as Flutter, a mobile application has been developed. Flutter is a self-contained cross-platform framework that compiles on all platforms. The introduction defines the goals and objectives of the study. The theoretical part analyzes the basic concepts of trade management automation. The design part provides a description of the development and architecture of the application. The conclusion summarizes the results and suggests directions for further research.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	7
ГЛАВА I. ПОНЯТИЕ МОБИЛЬНОЙ РАЗРАБОТКИ.....	12
1.1. Понятие автоматизации и ее значение.	12
1.2. Обзор существующих сервисов автоматизации публикаций.	13
1.3. Что такое Flutter.....	16
1.4. Обзор языка программирования Dart.	20
1.5. Основы Dart для разработки мобильных приложений.	23
1.6. Основы работы с фреймворком Flutter.....	33
ГЛАВА II. РАЗРАБОТКИ УПРАВЛЕНИЕ ТОРГОВЛИ В МАГАЗИНАХ ...	44
2.1. Создание технического задания проекта.	44
2.2. Выбор платформы для создания пользовательского интерфейса.	46
2.3. Настройка среды разработки.....	47
2.4. Создания пользовательского интерфейса.	51
2.5. Сборка проекта и тестирование функциональности.....	55
ГЛАВА III. ОФОРМЛЕНИЕ РЕЗУЛЬТАТОВ И ДОСТИЖЕНИЕ ПРОЕКТА	58
3.1. Сборка проекта и тестирование функциональности.....	58
3.2. Сохранение товаров в базу, работа с SQL базой	68
ГЛАВА IV. ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ	73
4.1 Работа с компьютерным оборудованием и анализ факторов риска	73
4.2. Пожарная безопасность	77
ЗАКЛЮЧЕНИЕ.....	84
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	85
ПРИЛОЖЕНИЕ	87

ВВЕДЕНИЕ

Широкое внедрение цифровых технологий способствует эффективности государственного и общественного управления, развитию социальной сферы, одним словом, кардинальному улучшению жизни людей

Президент Республики Узбекистан Ш.М.Мирзиёев

В мире сегодняшнего цифрового бизнеса мобильные приложения становятся ключевым инструментом для управления торговлей. Бизнесмены и компании активно используют их для удобного контроля за запасами, обработки заказов и взаимодействия с клиентами. При увеличении объема торговли возникает потребность в эффективных решениях для автоматизации и оптимизации процессов управления, чтобы сэкономить время и ресурсы.

Актуальность проблемы. В сфере управления торговлей, ручное управление процессами, такими как учет товаров, обработка заказов и взаимодействие с клиентами, может быть сложным и времязатратным. Каждый аспект требует внимательного контроля и соответствия определенным стандартам и правилам. Более того, необходимо учитывать оптимальное время для выполнения различных операций, чтобы обеспечить эффективность и удовлетворение клиентов.

Ручное управление не только увеличивает вероятность ошибок, но и требует значительных временных и трудовых затрат. В условиях жесткой конкуренции и стремительно меняющегося рынка, эффективное использование ресурсов становится критически важным. Здесь на помощь приходят современные технологии, в частности мобильные приложения, которые позволяют автоматизировать многие рутинные процессы.

Цель и задачи работы. Целью данного дипломного проекта является разработка мобильного приложения для управления торговлей, основанного на современных технологиях мобильной разработки. Основная задача заключается в создании инструмента, который обеспечит пользователям

простой и эффективный способ управления запасами, обработки заказов и взаимодействия с клиентами через мобильные устройства.

Для достижения поставленной цели необходимо решить следующие задачи:

Провести анализ существующих решений и определить их недостатки и преимущества.

Разработать архитектуру мобильного приложения, учитывая требования гибкости и масштабируемости.

Реализовать функционал для управления запасами, обработки заказов и взаимодействия с клиентами.

Обеспечить интеграцию с внешними сервисами для расширения функциональных возможностей приложения.

Провести тестирование приложения для выявления и устранения возможных ошибок.

Разработать рекомендации по дальнейшему развитию и улучшению приложения.

Значимость и преимущества работы. Значимость и преимущества разработки мобильного приложения для управления торговлей неоспоримы в современном бизнесе. Основные плюсы включают:

Экономия времени и ресурсов: Мобильное приложение позволяет пользователям оптимизировать процессы управления запасами и обработки заказов, освобождая их от необходимости заниматься этим вручную. Это также снижает риск ошибок, связанных с ручным вводом данных.

Улучшенное планирование: Приложение позволяет заранее планировать операции по управлению торговлей, что способствует более эффективному использованию времени и ресурсов. Это позволяет пользователям достигать более широкой аудитории и повышать эффективность бизнеса.

Расширение охвата аудитории: Мобильное приложение обеспечивает возможность эффективного взаимодействия с клиентами через различные

каналы связи, что позволяет расширить охват целевой аудитории и улучшить результативность маркетинговых усилий.

Увеличение точности учета: Автоматизация процессов учета позволяет значительно повысить точность данных, что в свою очередь улучшает качество аналитики и принятие управлеченческих решений.

Повышение лояльности клиентов: Современные мобильные приложения могут предложить клиентам дополнительные удобства, такие как персонализированные предложения, упрощенная система заказов и обратная связь, что способствует увеличению лояльности клиентов.

Обзор существующих решений. На рынке представлено множество систем для управления торговлей, таких как 1С, SAP, Microsoft Dynamics и другие. Эти системы предлагают широкий спектр функционала, однако, многие из них либо слишком сложны для малого и среднего бизнеса, либо требуют значительных финансовых вложений на этапе внедрения и поддержки.

Рассмотрение существующих решений показало, что наибольшую популярность среди малого бизнеса имеют облачные сервисы, которые предлагают гибкие тарифные планы и не требуют значительных капитальных вложений. Однако, многие из них не учитывают специфику локального рынка и не предоставляют достаточную гибкость для интеграции с другими системами.

Технологическая основа проекта. Выбор платформы Flutter для разработки мобильного приложения обусловлен рядом преимуществ:

Кроссплатформенность: Возможность создания приложений для iOS и Android из одного исходного кода.

Высокая производительность: Приложения на Flutter работают практически так же быстро, как и нативные приложения.

Богатый набор виджетов: Flutter предоставляет большой набор виджетов, которые позволяют создавать красивые и функциональные пользовательские интерфейсы.

Сообщество и поддержка: Большое и активное сообщество разработчиков, обширная документация и поддержка от Google делают Flutter отличным выбором для реализации проекта.

Разработка архитектуры приложения. Архитектура приложения была спроектирована с использованием шаблона проектирования BLoC (Business Logic Component), что обеспечивает разделение пользовательского интерфейса и бизнес логики, упрощает тестирование и поддержку кода. Основные компоненты архитектуры включают:

UI слой: Отвечает за отображение данных и взаимодействие с пользователем.

BLoC слой: Содержит бизнес логику и управляет состоянием приложения.

Data слой: Отвечает за доступ к данным, взаимодействие с API и локальное хранение данных.

Интеграция с внешними сервисами

Для расширения функциональности приложения были интегрированы различные внешние сервисы:

Google Maps: Для управления логистикой и отображения местоположения складов и торговых точек.

Stripe: Для обработки платежей и интеграции с банковскими системами.

Firebase: Для аутентификации пользователей, хранения данных и отправки push-уведомлений.

Безопасность и защита данных

Особое внимание было уделено вопросам безопасности. Реализована многоуровневая система защиты данных, включающая:

Шифрование данных на устройстве: Для защиты локально хранимых данных.

Шифрование данных при передаче: Использование протоколов HTTPS и TLS для защиты данных при передаче между клиентом и сервером.

Аутентификация и авторизация: Использование OAuth 2.0 и JWT для обеспечения безопасности доступа к API.

Тестирование и оптимизация. Приложение прошло серию строгих тестов, включая функциональное тестирование, тестирование производительности и юзабилити-тестирование. Это позволило не только убедиться в стабильности работы приложения, но и значительно улучшить пользовательский опыт за счет оптимизации интерфейса и ускорения реакции приложения на действия пользователя.

Заключение и перспективы развития. Разработанное мобильное приложение успешно решает задачи управления торговлей и предоставляет значительные преимущества для бизнеса за счет автоматизации ключевых процессов. В дальнейшем планируется расширение функциональности сервиса, включение интеллектуальных алгоритмов для прогнозирования спроса и анализа поведения покупателей, а также развитие модулей для интеграции с IoT устройствами для умного складского учета.

ГЛАВА I. ПОНЯТИЕ МОБИЛЬНОЙ РАЗРАБОТКИ

1.1. Понятие автоматизации и ее значение.

В современном информационном обществе автоматизация является важной составной частью различных отраслей деятельности. Этот процесс представляет собой замену ручных операций автоматизированными системами, способными выполнять задачи без участия человека. В контексте мобильного приложения для управления торговлей в магазинах, автоматизация означает использование технологий и инструментов для автоматической обработки различных операций, связанных с управлением торговлей. Значимость автоматизации в данной области проявляется в нескольких аспектах:

1. Экономия времени и ресурсов: Автоматизация позволяет существенно сократить время, затрачиваемое на выполнение рутинных операций в управлении торговлей. Вместо ручной обработки каждой операции, мобильное приложение может автоматически выполнять множество задач, освобождая время персонала для выполнения более важных задач.

2. Упрощение процессов: Автоматизация способствует стандартизации и упрощению процессов управления торговлей. Мобильное приложение может предоставить единый интерфейс для управления складскими запасами, заказами от поставщиков, а также взаимодействия с клиентами, что облегчает работу персонала и уменьшает вероятность ошибок.

3. Улучшение точности и надежности: Автоматизированные системы обладают более высокой точностью и надежностью в сравнении с ручными операциями. Мобильное приложение может выполнять расчеты, обработку данных и выполнение других операций без человеческого вмешательства, что снижает риск ошибок и исключает возможность человеческих промахов.

4. Планирование и оптимизация: Мобильное приложение для управления торговлей может помочь оптимизировать бизнес-процессы путем анализа данных и принятия обоснованных решений. Оно может предоставить

инструменты для анализа продаж, прогнозирования спроса, а также оптимизации распределения товаров по магазину и взаимодействия с клиентами.

5. Анализ результатов и принятие решений: Мобильное приложение может предоставить данные и аналитику о продажах, поведении клиентов и эффективности бизнес-процессов. Это поможет владельцам и менеджерам магазинов принимать обоснованные решения по улучшению работы бизнеса и повышению его эффективности.

6. Интеграция с другими системами: Мобильное приложение для управления торговлей может интегрироваться с другими системами, такими как системы учета, CRM и платежные системы. Это позволит автоматизировать обмен данными между различными системами, упростить бухгалтерские и административные процессы, а также обеспечить более эффективное взаимодействие с клиентами и поставщиками.

Таким образом, использование мобильного приложения для управления торговлей в магазинах позволяет существенно улучшить эффективность бизнес-процессов, повысить качество обслуживания клиентов и обеспечить конкурентное преимущество на рынке.

1.2. Обзор существующих сервисов автоматизации публикаций.

1. Bitrix24:

- Сайт: [Bitrix24](<https://www.bitrix24.ru/>).

- Bitrix24 предоставляет широкий спектр инструментов для автоматизации управления торговлей, включая CRM, управление задачами, отслеживание продаж, аналитику и многое другое. Платформа позволяет управлять клиентскими базами данных, сделками, заказами, интегрироваться с платежными системами и многое другое.

- Тарифы: Bitrix24 предлагает несколько тарифных планов, начиная от бесплатного, который подходит для небольших команд и предпринимателей, и заканчивая корпоративными планами с расширенным функционалом и дополнительной поддержкой.

- Преимущества: Широкий функционал, возможность интеграции с другими сервисами, удобный интерфейс, наличие мобильного приложения.

- Недостатки: Некоторые функции могут быть избыточны для малых компаний, требует времени на изучение и настройку.

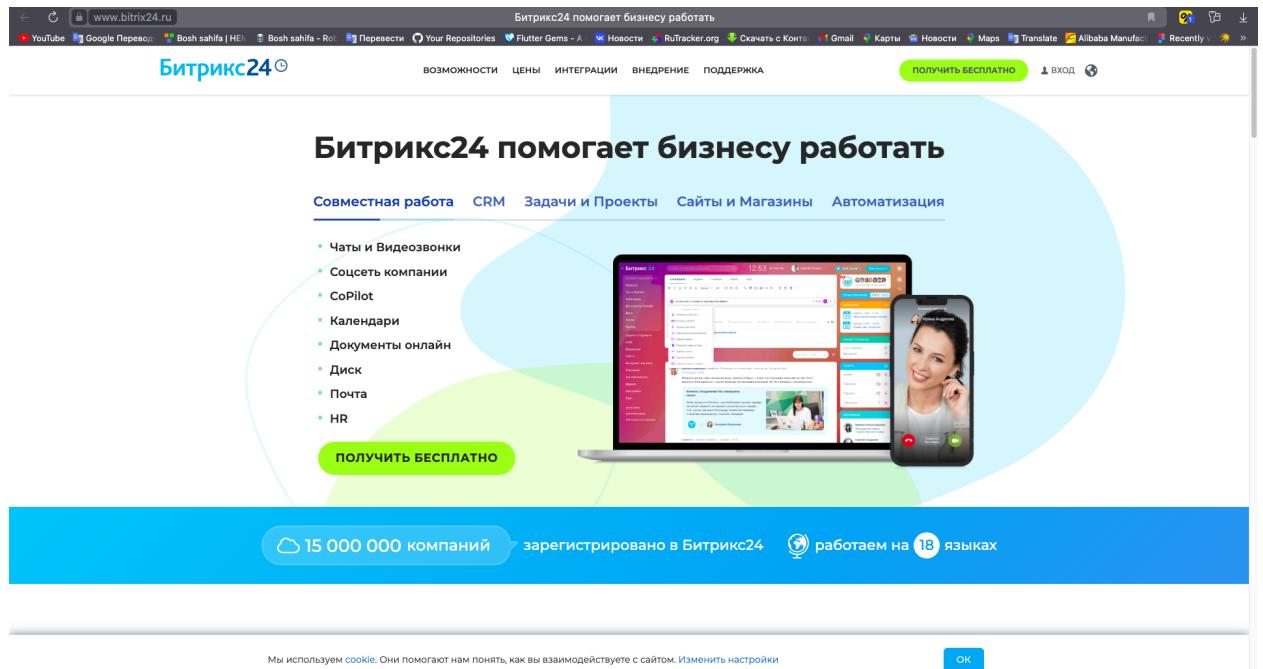


Рисунок 1.2.1 – Bitrix24

2. RetailCRM:

- Сайт: [RetailCRM](<https://retailcrm.ru/>).

- RetailCRM - это инструмент для автоматизации управления торговлей, специализирующийся на розничных продажах. Он предоставляет функции управления клиентской базой данных, учета складских запасов, обработки заказов, создания отчетов и аналитики. Система также интегрируется с популярными платежными и доставочными сервисами.

- Тарифы: RetailCRM предлагает несколько тарифных планов, включая бесплатный план с ограниченным функционалом и платные планы с расширенными возможностями и дополнительной поддержкой.

- Преимущества: Специализация на управлении розничными продажами, широкий функционал, аналитика и отчетность, возможность интеграции с внешними сервисами.

- Недостатки: Могут быть избыточные функции для малых компаний, некоторые пользователи отмечают некоторую сложность в настройке.

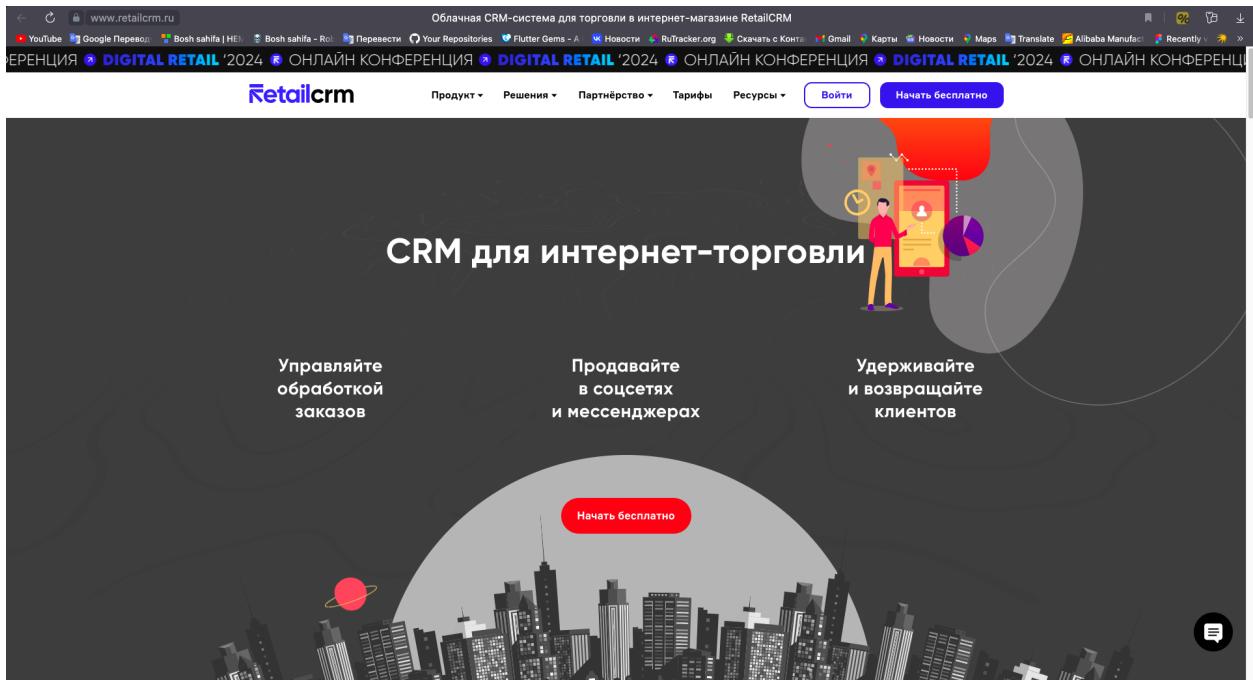


Рисунок 1.2.2 – RetailCRM

3. 1С:Управление торговлей:

- Сайт: [1С:Управление торговлей](<https://v8.1c.ru/trade/>).
- 1С:Управление торговлей - это программное обеспечение, предназначенное для автоматизации торговых предприятий различных масштабов. Оно включает в себя модули для управления продажами, складского учета, закупок, финансов и учета документов. Платформа также обеспечивает интеграцию с другими системами и онлайн-магазинами.
- Тарифы: Стоимость лицензии и условия использования 1С зависят от версии продукта, числа пользователей и модулей, которые включены в пакет. Она может варьироваться от одноразовой покупки до аренды по подписке.
- Преимущества: Широкие возможности по настройке, надежность, широкая распространность, возможность интеграции с другими программными продуктами.
- Недостатки: Необходимость внедрения и настройки, зависимость от квалифицированного персонала для поддержки, возможные ограничения в гибкости и масштабируемости в сравнении с более современными решениями.

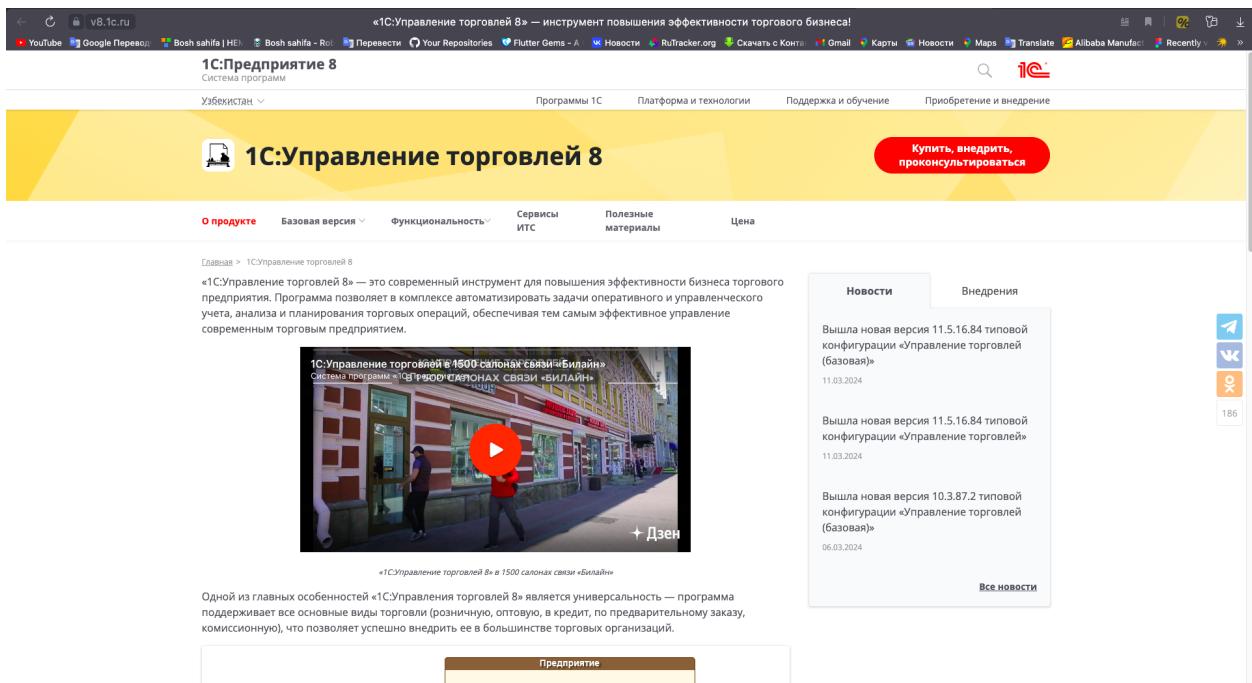


Рисунок 1.2.3 - 1C:Управление торговлей

Каждый из этих сервисов предлагает свои особенности и преимущества, подходящие для различных потребностей в управлении торговлей.

1.3. Что такое Flutter.

Flutter — это бесплатный фреймворк с открытым исходным кодом для разработки мобильных пользовательских интерфейсов, созданный Google и выпущенный в мае 2017 года. Если вкратце, он позволяет создавать нативные мобильные приложения только с одной кодовой базой. Это означает, что вы можете использовать один язык программирования и одну кодовую базу для создания 2 разных приложений (под iOS и Android).

Flutter состоит из 2 важных компонентов.

SDK (Software Development Kit) — набор инструментов, которые помогут при разработке приложения. В него входят средства для компиляции кода в нативный машинный код (код для iOS и Android).

Framework (UI-библиотека на основе виджетов) — коллекция многократно используемых элементов пользовательского интерфейса (кнопки, текстовые вводы, слайдеры и т.д.), которые можно настраивать в зависимости от персональных предпочтений. При разработке с помощью Flutter используется язык программирования Dart. Он был создан компанией

Google в октябре 2011 года и за это время претерпел значительные улучшения. Dart ориентирован на фронтенд-разработку и может использоваться для создания как мобильных, так и веб-приложений. Это типизированный язык объектного программирования, который можно сравнить с JavaScript по синтаксису.

Зачем изучать Flutter? Я выбрал несколько причин, почему мне нравится Flutter и почему я собираюсь использовать его в текущем году. Ниже я поделюсь своими соображениями.



Рисунок 1.3.1 – Flutter используют эти компании

Простота в освоении и использовании. Flutter — это современный фреймворк, и вы можете легко в этом убедиться! С его помощью гораздо проще создавать мобильные приложения. Если вы до этого использовали Java, Swift или React Native, вы заметите, что Flutter отличается от них. Лично мне не нравилось разрабатывать мобильные приложения, пока я не познакомился с Flutter. С ним можно создавать настоящие нативные приложения без огромного количества кода.

Быстрая компиляция, максимальная производительность
Благодаря Flutter можно изменять код и видеть результаты в режиме реального времени. Это называется “горячая перезагрузка”. Обновления самого приложения после сохранения требует совсем немного времени. Вам придется перезагружать приложение после внесения значительных изменений. Но если вы занимаетесь, например, дизайном и изменяете размер элемента, то это все будет происходить в режиме реального времени!

Идеально подходит для MVP. Если вы хотите как можно скорее показать свой продукт инвесторам, Flutter станет хорошим выбором. Вот 4 причины использовать этот фреймворк для создания MVP (минимально жизнеспособного продукта). Разработка мобильного приложения с помощью Flutter обходится дешевле, так как вам не нужно создавать и поддерживать 2 мобильных приложения (для iOS и Android).

Для создания MVP нужен всего 1 разработчик.

Этот фреймворк обладает высокой производительностью: вы не заметите разницы между нативным приложением и приложением на Flutter.

Широкий выбор привлекательных визуальных элементов: вы можете использовать виджеты, предоставляемые Flutter, и персонализировать их, чтобы создать ценный пользовательский интерфейс для клиентов (примеры приложений, созданных с помощью Flutter, вы найдете ниже).

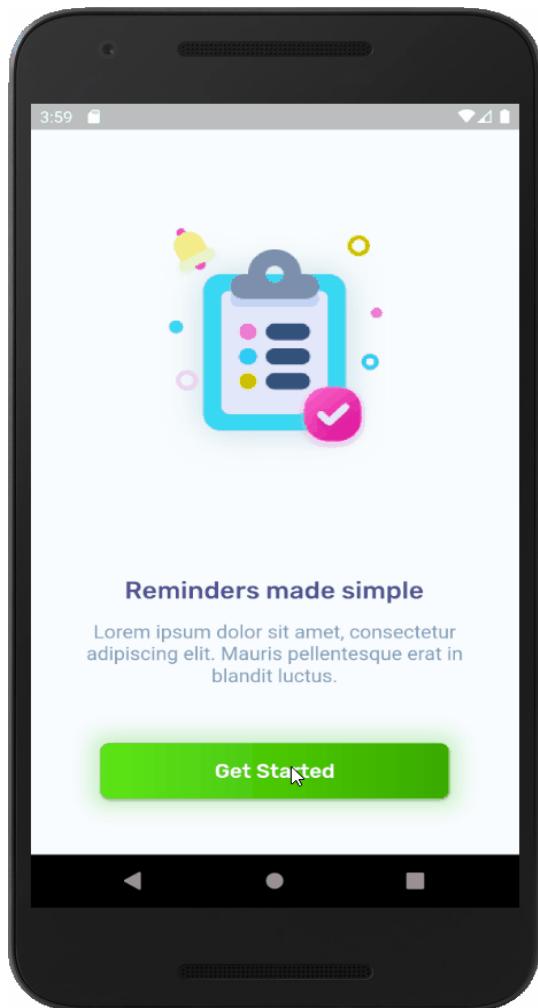


Рисунок 1.3.2 – Простое приложение

Подробная и доступная документация. Если речь идет о новой технологии, неплохо иметь под рукой хорошую документацию. Но так бывает не всегда! Вы можете многое узнать из документации Flutter. Там все очень подробно описано, и приведены простые примеры для базовых случаев использования. Каждый раз, когда у меня возникала проблема с каким-либо виджетом в коде, я обращался к документации и находил там ответ на свой вопрос.

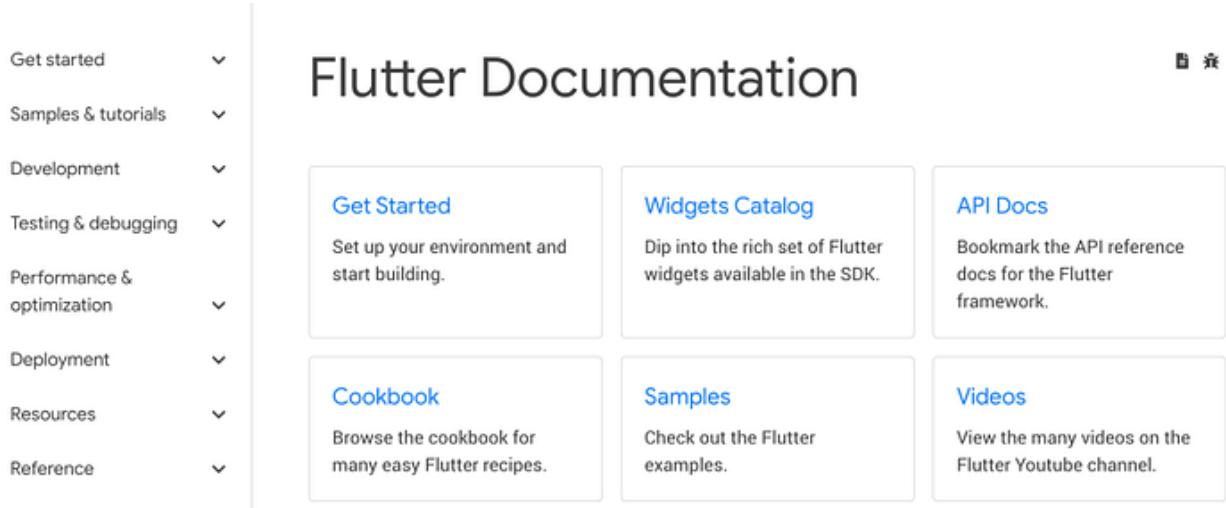


Рисунок 1.3.3 – Документация Flutter

Растущее сообщество. Flutter может похвастаться сильным комьюнити, и это только начало! Когда я начинал пользоваться Flutter, я сразу отправился на поиск тематических сообществ. К моему удивлению, я нашел значительное количество платформ для обмена информацией о Flutter.

Flutter Awesome. Потрясающий сайт, на котором собраны лучшие библиотеки и инструменты Flutter. Здесь ежедневно публикуются материалы с большим количеством примеров, шаблонов приложений, советов и т.д.

Awesome Flutter. Репозиторий на GitHub, связанный с Flutter Awesome, в котором вы найдете множество статей, видео, компонентов, утилит и т.д.

It's all widgets! Открытый список приложений, созданных с помощью Flutter.

Flutter Community. Блог на Medium, где вы можете найти тематические статьи, руководства по Flutter и многое другое.

Поддерживается Android Studio и VS Code. Flutter доступен в различных IDE. 2 основных редактора кода для разработки с использованием этой технологии: Android Studio (IntelliJ) и VS Code. Android Studio — полноценная программа, в которую уже все интегрировано. Для начала работы нужно загрузить плагины Flutter и Dart. VS Code — легкий инструмент, в котором все настраивается с помощью плагинов из маркетплейса.

1.4. Обзор языка программирования Dart.

Dart - ООП сценарный язык программирования, предназначенный для создания веб-приложений и мобильного ПО. Синтаксис Dart напоминает C и JavaScript. Одной из целей разработчиков языка было создание, по сути, улучшенной версии JS, которая эффективно решала бы практические задачи программирования. Корпорация Google впервые представила публике новую платформу разработки в сентябре 2011 года. Создатели задумывали Дарт как гибкий и производительный язык программирования общего назначения, который легко освоить при знании JavaScript.

Первая официальная стабильная сборка Dart вышла только в 2013 году. На данный момент язык активно обновляется. Актуальная версия платформы под номером 2.18.5 вышла в ноябре 2022. Сообщество Dart и в целом влияние сейчас растет как внутри, так и вне Google. Наибольшую известность Dart приобрел, став частью Flutter - фреймворка для мультиплатформенной разработки. Чаще всего на Flutter пишут мобильные приложения, но также на нем можно создавать веб-приложения и десктопное ПО. Но Дарт не ограничивается Flutter. На этом языке можно написать:

- серверные приложения;
- приложения командной строки (CLI);
- десктопное и мобильное программное обеспечение;

ПО для интернета вещей. В Google, к примеру, Dart используется для управления платформой Google Ads. По своей сути Дарт - универсальная платформа. Ее применение ограничено лишь логикой и удобством в каждом конкретном случае.

Особенности и преимущества:

Just-in-time (JIT) и Ahead-of-time (AOT) компиляция. Обычно исполняющее устройство обрабатывает программный код одним из двух способов. В первом случае компилятор переводит байт-код (промежуточный код) непосредственно в машинный код “на лету” (just-in-time). Во втором программа компилируется полностью от начала до конца. JIT ускоряет процесс разработки ПО, так как необходимые правки в код можно вносить фактически сразу. Правда, при непосредственном выполнении программы могут возникать “провисания” и лаги. АОТ занимает достаточно времени на компиляцию, но при этом уже скомпилированная программа работает более плавно и предсказуемо, и запускается быстрее. Dart совмещает оба вида обработки кода, пользуясь их преимуществами во время разработки и при выпуске готового продукта.

Промежуточная компиляция. Программу на Dart с помощью виртуальной машины без труда можно скомпилировать в JavaScript для выполнения в браузере. Это повышает кроссплатформенность кода.

Продвинутый сборщик мусора. Удаление мусора и эффективное распределение памяти особенно важно для приложений с быстро меняющимся пользовательским интерфейсом. Сборщик мусора, работающий на основе поколений объектов и избегающий излишних блокировок при выполнении кода, обеспечивает высокую частоту обновления экрана и плавность анимации.

Опциональные типы. В Dart можно как строго указывать типы, так и отдавать их определение на усмотрение компилятора. Такое совмещение строгой и динамической типизации позволяет, с одной стороны, ускорить написание кода, а с другой - избежать лишних ошибок там, где они действительно могли бы возникнуть.

Асинхронность. По своему устройству Dart - однопоточный язык. Если какая-то операция в потоке будет выполняться слишком долго, приложение может подвиснуть. Для предотвращения таких ситуаций в Dart

предусмотрено асинхронное выполнение операций. Благодаря фиче Event Loop обработка “тяжелой” части кода временно откладывается, и блокировки потока не происходит.

Быстрота и плавность работы конечного продукта. Приложения на Dart отлично работают при частоте обновления экрана в 60 FPS. Множество функций (в том числе упоминавшийся сборщик мусора) направлены на плавность переходов и отсутствие зависаний. Фреймворк Flutter считается лучшим кроссплатформенным решением в этом плане, не уступая по всем показателям большинству нативных разработок.

Открытый код. Дарт - продукт с открытым исходным кодом. Такой подход способствует развитию языка и заинтересованности сообщества в его улучшении. Владелец платформы, компания Google, всегда учитывает мнение пользователей при выпуске обновлений своей платформы.

Простота освоения. При знании JavaScript или C++ разработчик может освоить Dart за очень короткое время. Синтаксис Dart может показаться более строгим, чем в Swift или Kotlin, но не критически.

Недостатки. Если бы язык Dart был таким совершенным, как о нём говорят его разработчики, то о JavaScript и TypeScript мы бы стали быстро забывать. Но пока это не так. Вот главные претензии:

Малое сообщество. При всех плюсах самого языка, Google каким-то непостижимым образом не уделил должного внимания простым разработчикам. Нет, формально всё хорошо: недавно в Мюнхене прошла крупная конференция «Dart Dev Summit», а официальный сайт располагает всей необходимой информацией, как для новичка, так и для профессионала. Но вот сообщество разработчиков оставляет желать лучшего. Так, например, если зайти на официальный форум, то можно заметить, что регулярность задаваемых вопросов – 2 в месяц, а количество просмотров каждого не превышает сотни. Надо ли говорить о том, что в этом Dart не конкурент JavaScript?

Проблемы JavaScript преувеличены. Если покопаться в интернет-истории, то можно найти немало adeptов JavaScript, которые изначально были настроены негативно по отношению к детищу Google. Как тогда, так и сейчас, основная претензия сводится к тому, что будь JavaScript настолько ограниченным языком, то он не сыскал бы славу универсального инструмента, как для простых скриптов, так и для сложных приложений. Более того, в прошлом году даже в Google перестали видеть в JavaScript конкурента и всерьёз занялись улучшением компилятора кода, но, как оказалось, лишь на время: «Dart Dev Summit 2016» ознаменовал возвращение Dart как полноценного игрока.

1.5. Основы Dart для разработки мобильных приложений.

Dart – это объектно-ориентированный язык программирования, разработанный компанией Google. Он используется для создания веб-приложений, мобильных приложений и серверных приложений.

Основные особенности языка Dart:

Статическая типизация: Dart поддерживает статическую типизацию, что позволяет выявлять ошибки на этапе компиляции.

Компиляция в JavaScript: Dart код может быть скомпилирован в JavaScript, что позволяет запускать Dart приложения в любом современном браузере.

Гибкая синтаксическая структура: Dart имеет простой и понятный синтаксис, который легко читать и писать.

Поддержка асинхронного программирования: Dart предоставляет мощные инструменты для работы с асинхронными операциями, такими как сетевые запросы и обработка событий.

Переменные и типы данных. В Dart есть несколько типов данных, включая числа, строки, булевые значения, списки, карты и классы. Переменные в Dart могут быть объявлены с явным указанием типа или без него.

Условные операторы и циклы. Условные операторы, такие как if-else и switch, позволяют выполнять различные действия в зависимости от условий. Циклы, такие как for и while, позволяют выполнять повторяющиеся действия.

Функции и методы. Функции в Dart позволяют группировать код для выполнения определенной задачи. Они могут принимать аргументы и возвращать значения. Методы – это функции, которые принадлежат определенному классу.

Классы и объекты. Классы в Dart позволяют определять объекты с определенными свойствами и методами. Объекты являются экземплярами класса и могут быть созданы с использованием конструктора класса.

Модули и пакеты. Модули и пакеты в Dart позволяют организовывать код в логические блоки и повторно использовать его в разных проектах. Модули – это файлы с кодом, а пакеты – это набор связанных модулей.

Асинхронное программирование. Асинхронное программирование в Dart позволяет выполнять операции, которые занимают много времени, без блокировки основного потока выполнения. Dart предоставляет ключевые слова async и await для работы с асинхронными операциями.

Отладка и тестирование. Dart предоставляет инструменты для отладки и тестирования кода, такие как отладчик и фреймворк для модульного тестирования. Они помогают выявлять и исправлять ошибки в коде.

Примеры использования Dart. Dart может быть использован для разработки различных типов приложений, включая веб-приложения, мобильные приложения и серверные приложения. Он имеет богатую экосистему инструментов и библиотек, которые облегчают разработку и ускоряют процесс создания приложений.

Переменные и типы данных. Переменные – это именованные области памяти, которые используются для хранения данных в программе. В языке Dart переменные могут быть объявлены с помощью ключевого слова var или с указанием конкретного типа данных.

Типы данных в Dart. В Dart есть несколько встроенных типов данных:

int – целочисленный тип данных, который представляет целые числа.

double – тип данных с плавающей точкой, который представляет числа с плавающей точкой.

bool – логический тип данных, который представляет значения true или false.

String – тип данных, который представляет строки символов.

List – тип данных, который представляет упорядоченный список элементов.

Map – тип данных, который представляет ассоциативный массив, состоящий из пар ключ-значение.

Объявление переменных. Переменные в Dart могут быть объявлены с помощью ключевого слова var или с указанием конкретного типа данных. Например:

```
var age = 25; // переменная age имеет тип int  
double price = 99; // переменная price имеет тип double  
bool isStudent = true; // переменная isStudent имеет тип bool  
String name = "John"; // переменная name имеет тип String  
List<int> numbers = [1, 2, 3, 4, 5]; // переменная numbers имеет тип  
List<int>  
Map<String, int> scores = {"Math": 90, "Science": 85}; // переменная scores  
имеет тип Map<String, int>
```

Присваивание значений переменным. Значения переменных могут быть присвоены с помощью оператора присваивания (=). Например:

```
int x = 10;  
double y = 14;  
bool isReady = true;  
String message = "Hello, world!";
```

Изменение значений переменных. Значения переменных могут быть изменены путем присваивания нового значения. Например:

```
int x = 10;
```

```
x = 20; // значение переменной x изменено на 20
```

Константы. Константы – это переменные, значения которых не могут быть изменены после инициализации. В Dart константы объявляются с помощью ключевого слова `final` или `const`. Например:

```
final int age = 25; // константа age имеет тип int  
const double pi = 14; // константа pi имеет тип double
```

Константы могут быть инициализированы только один раз и не могут быть изменены после этого.

Условные операторы и циклы. Условные операторы и циклы – это инструменты, которые позволяют программе принимать решения и выполнять повторяющиеся действия в зависимости от определенных условий.

Условные операторы. Условные операторы позволяют программе выполнять различные действия в зависимости от условий. В языке Dart есть несколько условных операторов:

Оператор if. Оператор `if` позволяет выполнить блок кода, если указанное условие истинно. Например:

```
int x = 10;  
  
if (x > 5) {  
  
    print("x больше 5");  
  
}
```

Если значение переменной `x` больше 5, то будет выведено сообщение “`x` больше 5”.

Оператор if-else. Оператор `if-else` позволяет выполнить один блок кода, если условие истинно, и другой блок кода, если условие ложно. Например:

```
int x = 10;  
  
if (x > 5) {  
  
    print("x больше 5");  
  
} else {  
  
    print("x меньше или равно 5");  
  
}
```

Если значение переменной x больше 5, то будет выведено сообщение “x больше 5”, иначе будет выведено сообщение “x меньше или равно 5”.

Оператор switch. Оператор switch позволяет выполнить различные действия в зависимости от значения переменной. Например:

```
int day = 3;  
switch (day) {  
    case 1:  
        print("Понедельник");  
        break;  
    case 2:  
        print("Вторник");  
        break;  
    case 3:  
        print("Среда");  
        break;  
    default:  
        print("Другой день");  
        break;  
}
```

В этом примере будет выведено сообщение “Среда”, так как значение переменной day равно

Циклы. Циклы позволяют программе выполнять повторяющиеся действия. В языке Dart есть несколько типов циклов:

Цикл while. Цикл while выполняет блок кода, пока указанное условие истинно. Например:

```
int i = 0;  
while (i < 5) {  
    print(i);  
    i++;  
}
```

В этом примере будут выведены числа от 0 до

Цикл do-while. Цикл do-while выполняет блок кода, а затем проверяет условие. Если условие истинно, цикл повторяется. Например:

```
int i = 0;  
do {  
    print(i);  
    i++;  
} while (i < 5);
```

В этом примере также будут выведены числа от 0 до

Цикл for. Цикл for выполняет блок кода определенное количество раз. Например:

```
for (int i = 0; i < 5; i++) {  
    print(i);  
}
```

В этом примере также будут выведены числа от 0 до

Цикл for-in. Цикл for-in используется для перебора элементов в коллекции, такой как список или массив. Например:

```
List<int> numbers = [1, 2, 3, 4, 5];  
for (int number in numbers) {  
    print(number);  
}
```

В этом примере будут выведены числа 1, 2, 3, 4,

Условные операторы и циклы являются важными инструментами в программировании, позволяющими создавать более сложные и гибкие программы.

Функции и методы. Функции и методы являются основными строительными блоками программирования. Они позволяют группировать код в отдельные блоки, которые могут быть вызваны и использованы в разных частях программы.

Функции. Функция - это блок кода, который выполняет определенную задачу. Она может принимать аргументы (входные данные) и возвращать результат (выходные данные). Функции могут быть определены внутри других функций или в глобальной области видимости.

Пример определения функции:

```
int sum(int a, int b) {  
    return a + b;  
}
```

В этом примере функция `sum` принимает два аргумента `a` и `b` типа `int` и возвращает их сумму.

Функции могут быть вызваны в других частях программы, их результаты могут быть сохранены в переменных или использованы непосредственно в других выражениях.

Пример вызова функции:

```
int result = sum(2, 3);  
print(result); // Выведет 5
```

Методы. Метод - это функция, которая определена внутри класса или объекта. Он может иметь доступ к свойствам и методам этого класса или объекта.

Пример определения метода:

```
class Person {  
    String name;  
    void sayHello() {  
        print("Hello, my name is  
        *** QuickLaTeX cannot compile formula:  
        name.");  
    }  
}
```

В этом примере класс `Person` имеет метод `sayHello`, который выводит приветствие с именем объекта.

Методы могут быть вызваны на объектах этого класса:

```
Person person = Person();
person.name = "John";
person.sayHello(); // Выведет "Hello, my name is John."
```

Функции и методы являются важными инструментами в программировании, позволяющими создавать модульный и переиспользуемый код. Классы и объекты. В программировании классы и объекты являются основными строительными блоками объектно-ориентированного подхода. Класс - это шаблон или формальное описание, которое определяет свойства и методы объекта. Объект - это экземпляр класса, который содержит конкретные значения свойств и может вызывать методы класса. Определение класса. Класс определяется с помощью ключевого слова **class**, за которым следует имя класса. Внутри класса можно определить свойства и методы.

Пример определения класса:

```
Person {
    String name;
    int age;
    void sayHello() {
        print("Hello, my name is $name and I am $age years old");
    },
}
```

В этом примере класс Person имеет два свойства - name и age, а также метод sayHello, который выводит приветствие с именем и возрастом объекта.

Создание объекта. Объект создается с помощью ключевого слова **new**, за которым следует имя класса и аргументы для конструктора класса (если он определен).

Пример создания объекта:

```
person = Person();
person.name = "John";
```

```
person.age = 25;  
person.sayHello(); // Выведет "Hello, my name is John and I am 25 years  
old."
```

В этом примере создается объект person класса Person и устанавливаются значения его свойств. Затем вызывается метод sayHello объекта, который выводит приветствие с именем и возрастом.

Классы и объекты позволяют организовывать код в логические блоки, упрощают его понимание и обеспечивают возможность повторного использования кода.

Модули и пакеты. В языке Dart модули и пакеты используются для организации и структурирования кода. Они позволяют разделить код на логические блоки, упростить его понимание и обеспечить возможность повторного использования.

Модули. Модуль - это файл с расширением .dart, который содержит определения классов, функций, переменных и других элементов кода. Модуль может быть использован в других модулях с помощью ключевого слова import. Пример импорта модуля:

```
import 'module.dart';
```

В этом примере модуль module.dart импортируется в текущий модуль. Теперь все определения из модуля module.dart доступны в текущем модуле

Пакеты. Пакет - это коллекция связанных модулей, которые объединены вместе для облегчения их использования. Пакеты могут содержать модули, а также другие пакеты. В языке Dart пакеты организуются с помощью файла pubspec.yaml. В этом файле указываются зависимости пакета, а также другие настройки.

Пример файла pubspec.yaml:

```
name: my_package  
version: 0  
dependencies:  
  package1: ^0
```

```
package2: ^0
```

В этом примере пакет my_package имеет версию 0 и зависит от пакетов package1 версии 0 и package2 версии

Пакеты могут быть опубликованы в репозитории пакетов Dart и установлены с помощью менеджера пакетов Dart, такого как pub или flutter pub.

Использование пакетов и модулей позволяет разделить код на множество файлов, упростить его организацию и повторное использование, а также улучшить поддержку и сопровождение кода.

Асинхронное программирование - это подход к написанию программ, который позволяет выполнять несколько задач одновременно и эффективно использовать ресурсы компьютера.

Понятие асинхронности. В традиционном синхронном программировании задачи выполняются последовательно, одна за другой. Каждая задача должна завершиться, прежде чем следующая может быть выполнена. Это может привести к блокировке программы, если одна из задач занимает много времени.

В асинхронном программировании задачи выполняются параллельно или в фоновом режиме, без блокировки основного потока выполнения. Это позволяет программе продолжать работу, пока выполняются другие задачи.

Преимущества асинхронного программирования. Асинхронное программирование имеет несколько преимуществ:

Улучшение отзывчивости: Асинхронные задачи выполняются в фоновом режиме, что позволяет программе оставаться отзывчивой и отвечать на пользовательские действия.

Эффективное использование ресурсов: Асинхронные задачи позволяют эффективно использовать ресурсы компьютера, такие как процессорное время и сетевые соединения.

Улучшение производительности: Асинхронное программирование позволяет параллельно выполнять несколько задач, что может улучшить производительность программы.

Механизмы асинхронного программирования в Dart. В языке Dart есть несколько механизмов для реализации асинхронного программирования:

Future и async/await: Future - это объект, который представляет результат асинхронной операции. Ключевые слова `async` и `await` используются для написания асинхронного кода в синхронном стиле.

Stream и StreamBuilder: Stream - это последовательность асинхронных событий. StreamBuilder позволяет реагировать на изменения в потоке и обновлять пользовательский интерфейс.

Isolate: Isolate - это легковесный процесс, который выполняется параллельно с основным потоком выполнения. Isolate позволяет выполнять вычислительно интенсивные задачи без блокировки основного потока.

Асинхронное программирование является важной темой в разработке программного обеспечения. Понимание его концепций и механизмов поможет вам создавать более отзывчивые и эффективные программы.

1.6. Основы работы с фреймворком Flutter.

Flutter - это современный фреймворк для разработки мобильных приложений, созданный компанией Google. Его история начинается в 2015 году, когда команда разработчиков в Google начала проект под кодовым названием "Sky". Их целью было создание инновационного инструмента для разработки интерфейсов, который бы обеспечивал высокую производительность и кросс-платформенную совместимость.

Однако ранние версии Flutter не сразу нашли своего пользователя. В течение нескольких лет проект проходил через многочисленные изменения и улучшения, а команда разработчиков трудилась над его совершенствованием.

Переломным моментом стала презентация Flutter на конференции Dart Developer Summit в октябре 2017 года. Здесь Google представила Flutter как полноценный инструмент для создания красивых и высокопроизводительных

мобильных приложений на платформах Android и iOS. Ключевыми особенностями Flutter были гибкий UI-фреймворк, называемый "Widget", а также Hot Reload, позволяющий разработчикам мгновенно видеть изменения в своем приложении без перезагрузки приложения.

С момента своего официального выпуска в 2018 году Flutter быстро завоевал популярность среди разработчиков. Его активно использовали как для создания небольших приложений, так и для разработки масштабных проектов мирового уровня.

Сегодня Flutter продолжает развиваться и расширять свой функционал. Он активно поддерживается сообществом разработчиков и становится все более востребованным инструментом в индустрии мобильной разработки.

Flutter состоит из нескольких ключевых компонентов и инструментов, которые позволяют разработчикам создавать кросс-платформенные мобильные приложения:

1. Фреймворк Flutter: Основа Flutter - это фреймворк, который предоставляет инструменты и библиотеки для разработки мобильных приложений. Фреймворк включает в себя гибкие и мощные виджеты для построения пользовательского интерфейса, систему макетов, механизмы управления состоянием и другие компоненты.

2. Язык программирования Dart: Flutter использует язык программирования Dart, который разработан Google. Dart предлагает современные возможности, такие как статическая типизация, асинхронное программирование и синтаксический сахар для удобства разработки.

3. Виджеты: Одна из ключевых концепций Flutter - это виджеты. Виджеты - это строительные блоки пользовательского интерфейса в Flutter. Они могут быть как простыми элементами, такими как текстовые поля или кнопки, так и более сложными компоновками, такими как списки или сетки.

4. Горячая перезагрузка (Hot Reload): Это одна из самых мощных возможностей Flutter. Горячая перезагрузка позволяет мгновенно видеть

результаты внесенных изменений в код приложения без необходимости перезапуска всего приложения.

5. Инструменты разработки: Flutter предоставляет разнообразные инструменты для разработки, включая командную строку Flutter CLI, интегрированные среды разработки (IDE) такие как Android Studio, IntelliJ IDEA и Visual Studio Code, а также мощные инструменты для отладки.

6. Material Design и Cupertino Widgets: Flutter предоставляет виджеты, которые соответствуют стандартам дизайна Material Design (для Android) и Cupertino (для iOS), что позволяет создавать красивые и адаптивные интерфейсы для обеих платформ.

Приложения запускаются на основе комбинации движка рендеринга (построенного на C++) и Flutter (построенного на Dart). Все файлы, сгенерированные таким образом, присоединяются к каждому приложению и программному обеспечению сборки SDK для конкретной платформы.

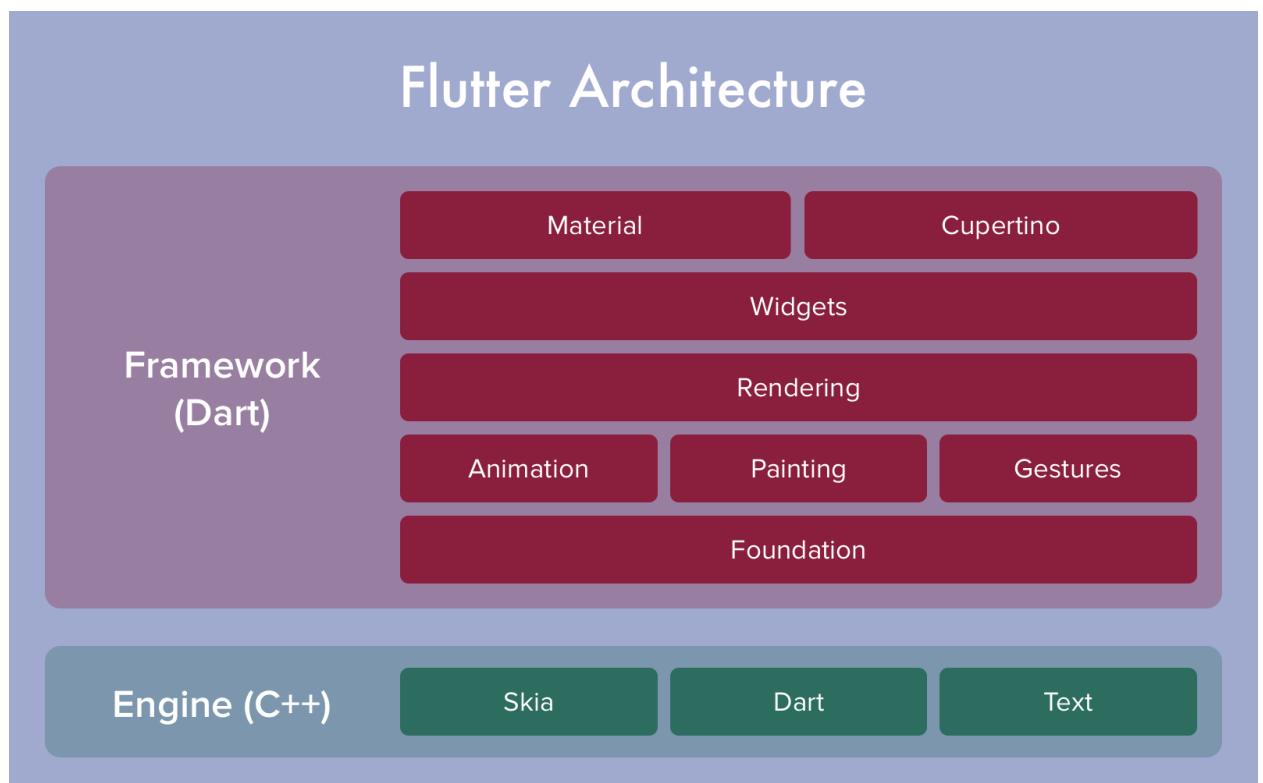


Рисунок 1.6.1 – Архитектура Flutter

Это как разработка игр: игра не выделяет своего фреймворка, а функциональность осуществляется игровым движком. То же самое и с

программным обеспечением Flutter — все приложения, основанные на Flutter SDK, заменяют части собственных фреймворков элементами Flutter.

Ещё один плюс Flutter — он ориентирован на Material Design и предоставляет множество возможностей для работы с ним. Google также использует Flutter для разработки пользовательского интерфейса своей новой системы Fuchsia.

Установка. Так как Flutter ещё в процессе разработки и постоянно обновляется, процесс установки со временем может поменяться. Актуальную инструкцию по установке можно найти на сайте Flutter.

Мы будем пользоваться версией 0.0.20+.alpha. (*Прим.перев.: на данный момент установка возможна только под Mac и Linux (64-bit)*)

Шаг 1. Клонирование

Клонируйте ветку alpha из репозитория Flutter при помощи Git (SourceTree, Github Desktop...) и добавьте директорию bin в PATH.

```
$ git clone https://github.com/flutter/flutter.git -b alpha  
$ export PATH=`pwd`/flutter/bin:$PATH
```

Шаг 2. Проверка зависимостей

Запустите Flutter doctor, чтобы установить все необходимые зависимости.

```
$ flutter doctor
```

Шаг 3. Установка платформ

Дальше мы установим платформы для разработки. Мы можем установить обе или ограничиться одной, для которой хотим написать приложение.

В случае с Android необходимо установить Android SDK. Можете просто установить Android Studio, SDK будет в комплекте. В случае, если Android Studio установлена не в директорию по умолчанию, необходимо добавить переменную ANDROID_HOME в PATH, указав новое расположение, куда был установлен SDK.

В случае с iOS необходим xCode версии 7.2 или выше. Для запуска приложений на физическом устройстве необходим дополнительный инструмент. Его можно установить при помощи Homebrew.

```
$ brew tap flutter/flutter  
$ brew install ideviceinstaller ios-deploy
```

Шаг 4. Конфигурация Atom

Рекомендуется использовать текстовый редактор Atom с установленными плагинами Flutter и Dart.

Установка плагина Flutter для Atom:

Запустите Atom.

Packages > Settings View > Install Packages/Themes.

Напишите в поле Install Packages слово ‘flutter’, затем нажмите кнопку Packages.

Выберите Flutter и установите.

Откройте Packages > Flutter > Package Settings и выставьте в FLUTTER_ROOT путь, куда был склонирован Flutter SDK.

Затем Packages > Dart > Package Settings и выставьте переменную с расположением dart sdk, обычно это bin/cache/dart-sdk в директории Flutter.

Если у вас Mac, запустите Atom > Install Shell Commands чтобы установить shell-команды. И напоследок запустите ещё раз Flutter doctor, чтобы удостовериться, что всё в порядке. Вывод из консоли ниже показывает, что процесс установки успешен, но среда iOS ещё не отвечает всем необходимым требованиям.

[✓] Flutter (on Mac OS, channel alpha)

- Flutter at /Users/XensS/dev-dart/flutter-sdk
- Framework revision 9a0a0d9903 (5 days ago), engine revision f8d80c4617

[✓] Android toolchain — develop for Android devices (Android SDK 24.0.1)

- Android SDK at /Users/XensS/Library/Android/sdk
- Platform android-N, build-tools 24.0.1

- Java(TM) SE Runtime Environment (build 1.8.0_25-b17)

[✓] iOS toolchain — develop for iOS devices (Xcode 6.4)

- XCode at /Applications/Xcode.app/Contents/Developer

- Xcode 6.4, Build version 6E35b

✗ Flutter requires a minimum XCode version of 7.0.0.

Download the latest version or update via the Mac App Store.

✗ ideviceinstaller not available; this is used to discover connected iOS devices.

Install via ‘brew install ideviceinstaller’.

✗ ios-deploy not available; this is used to deploy to connected iOS devices.

Install via ‘brew install ios-deploy’.

[✓] Atom — a lightweight development environment for Flutter

- flutter plugin version 0.2.4

- dartlang plugin version 0.6.37

Первые шаги (Пишем Hello World!)

Давайте создадим простенькое приложение и посмотрим Flutter в действии. В последующих статьях примеры будут куда сложнее и увлекательнее

Запустите Packages → Flutter → create new Flutter Project. В директории lib есть файл main.dart, откройте его и сотрите весь код.

Выполнение кода Dart начинается с функции main, которая должна быть включена в файл main.dart.

```
void main() { }
```

Теперь импортируем библиотеку material, она предоставляет нам функцию для запуска приложений.

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

Эта функция называется runApp и принимает виджет (Widget) в качестве параметра. Виджет можно сравнить с представлением (View) в Android или iOS, чтобы иметь общее представление, но, само собой, между ними есть и отличия. То есть в Flutter весь интерфейс строится на использовании виджетов

и весь код пишется на Dart. Например в Android надо было бы использовать XML для описания представлений.

Начнём с того, что выведем при помощи виджета Text произвольный текст.

```
import 'package:flutter/material.dart';
void main() {
  runApp(
    new Text("Hello World")
  );
}
```

Теперь запускаем приложение через Atom.

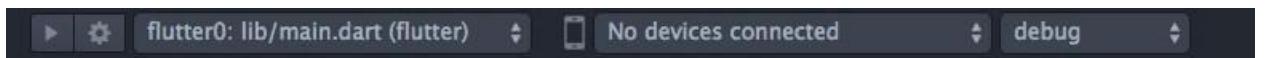


Рисунок 1.6.2 – Страна состояния

Как видно, текст появился за статус-баром. Так произошло потому, что



Рисунок 1.6.3 – Вид с надписью “Hello world”

туда установлены координаты Flutter (0,0).

Добавляется отступ, чтобы исправить это. Поскольку пользовательский интерфейс Flutter строится на виджетах, отступы тоже будут виджетом. Возможно, для людей с опытом разработки на Android и iOS (где отступы всего лишь свойства представления) это звучит дико. Нужно добавить виджет Padding и указать виджет Text как дочерний элемент.

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

void main() {
  runApp(ст
    new Padding(
      padding: const EdgeInsets.only(top: 24.0),
      child: new Text("Hello, World")
    )
  );
}
```

Ваш текст описывает использование виджета Padding во Flutter с использованием объекта EdgeInsets для создания отступа, и упоминает различия в использовании констант между Java и Dart. Вот расширенный и удвоенный вариант этого текста.

В примере выше мы создали виджет Padding, который применяет отступ в 24 единицы сверху, используя для этого объект EdgeInsets. Внутри этого виджета Padding размещён дочерний элемент — виджет Text. Когда вы запустите приложение, увидите, что текст располагается ниже из-за указанного отступа. Это позволяет более гибко управлять расположением элементов в пользовательском интерфейсе.

Примечание: если вы знакомы с языком программирования Java, то вам будет интересно знать, что использование const EdgeInsets.only(top: 24.0) в Dart эквивалентно вызову конструктора EdgeInsets, который создаёт экземпляр объекта. Однако, в отличие от Java, этот объект является константой

времени компиляции, что обеспечивает определённые преимущества, такие как улучшенная производительность и меньшее потребление памяти.

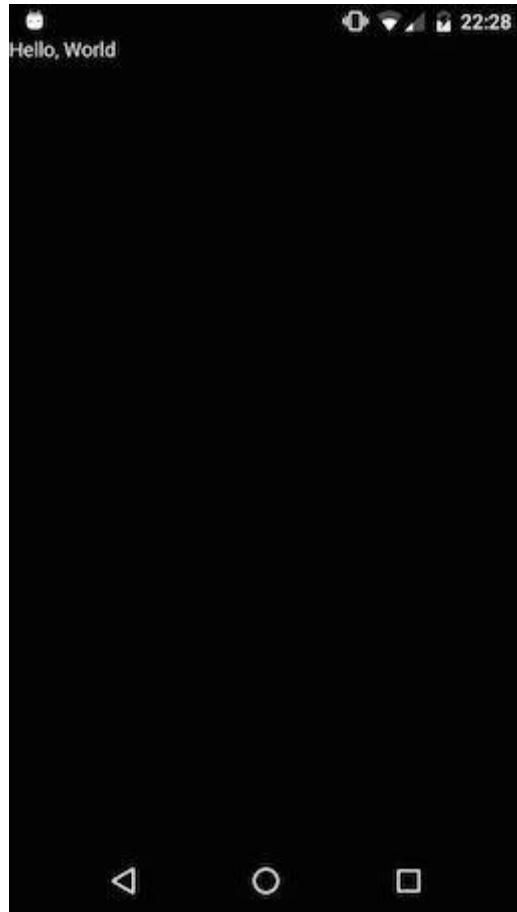


Рисунок 1.6.4 – Исправленный вид “Hello World”

Воспользуемся виджетом Center, чтобы разместить текст в центре экрана.

```
import 'package:flutter/material.dart';
void main() {
  runApp(
    new Center(
      child: new Text("Hello, World")
    )
);
}
```

Оба виджета, Padding и Center, предоставляют атрибут, называемый child, используемый для указания дочернего элемента. На самом деле это одна

из особенностей, делающих Flutter таким мощным инструментом. Каждый виджет может иметь дочерние элементы, благодаря чему одни виджеты могут быть вложены в другие виджеты. Так, например, Text может быть вложен в Padding, который будет вложен в Center.



Рисунок 1.6.5 – Текст в центре экрана

```
import 'package:flutter/material.dart';
void main() {
  runApp(
    new Center(
      child: new Padding(
        padding: const EdgeInsets.only(left: 128.0),
        child: new Text("Hello, World")
      ),
    ),
  ),
}
```

```
 );  
 }
```



Рисунок 1.6.6 – Центрированный но отступом с лева

ГЛАВА II. РАЗРАБОТКИ УПРАВЛЕНИЕ ТОРГОВЛИ В МАГАЗИНАХ

2.1. Создание технического задания проекта.

Цель проекта – разработать мобильное приложение "управление торговли в магазинах" для автоматизации торговли.

Описание функциональности проекта. Проект должен состоять из следующих функциональных блоков:

1. Введение:

- Цель: Создать программное обеспечение для учета и управления продажами.
- Область применения: Программа предназначена для использования в точках продаж различных предприятий.

2. Основные функции:

- Ввод информации о продажах, включая сумму, скидку, комментарий, сотрудника и тип оплаты.
- Управление товарным ассортиментом: добавление новых товаров, просмотр информации о товарах (название, изображение, остаток на складе, цена).
- Оформление корзины покупок с возможностью просмотра списка товаров, общей суммы, комментария и удаления товаров.
- История продаж и возвратов: возможность поиска по проданным товарам, открытым сменам, просмотр отчета о проданных товарах и возвратах.
- Формирование чека с указанием суммы покупки, цены без скидки и скидки, типа оплаты, имени покупателя и кассира.
- Управление сменами: открытие и закрытие смен, добавление денег в кассу, выплата средств, регистрация продаж и возвратов, учет выручки и остатка в кассе.

3. Требования к интерфейсу пользователя:

- Интерфейс должен быть интуитивно понятным и легким в использовании.

- Должна быть реализована возможность сканирования штрих-кодов для быстрого добавления товаров.

- Пользователь должен иметь доступ к основным функциям программы через меню или кнопки.

4. Требования к базе данных:

- База данных должна хранить информацию о товарах, сотрудниках, сменах, продажах и возвратах.

- Должна обеспечиваться безопасность и целостность данных.

- Доступ к базе данных должен осуществляться через защищенное подключение.

5. Требования к отчетности:

- Программа должна предоставлять возможность генерации отчетов о продажах за определенный период, сменный отчет, отчет о возвратах.

- Отчеты должны быть представлены в удобочитаемом формате и содержать необходимую информацию для анализа и контроля продаж.

6. Требования к безопасности:

- Должны быть предусмотрены механизмы аутентификации и авторизации пользователей.

- Предусмотрены меры по защите данных от несанкционированного доступа и искажения.

- Все операции с данными должны быть логируемыми для последующего аудита.

7. Требования к поддержке:

- Предусмотреть возможность обновления программного обеспечения.

- Обеспечить техническую поддержку и обучение персонала по использованию программы.

8. Заключение:

- Разработанное программное обеспечение должно соответствовать всем требованиям и быть готовым к внедрению в работу.

Предлагаемый стек технологий. Dart - это высокоуровневый язык программирования, который работает в интерпретируемой среде выполнения. Его синтаксис прост и легко читаем, что делает его привлекательным выбором для разработчиков. Dart широко применяется во многих областях, включая веб-разработку, мобильную разработку, а также разработку серверных приложений.

Одной из основных областей применения Dart является веб-разработка. Благодаря фреймворку Flutter, который основан на Dart, разработчики могут создавать красивые и высокопроизводительные мобильные приложения для платформ Android и iOS, а также веб-приложения и даже десктопные приложения. Экосистема Dart также богата библиотеками и фреймворками, что обеспечивает разработчикам широкие возможности для создания различных типов приложений. Сочетание простого синтаксиса, высокой производительности и широкого спектра инструментов делает Dart привлекательным выбором для множества разработчиков.

Flutter - это мощный и гибкий фреймворк для разработки кроссплатформенных мобильных приложений, созданный на языке программирования Dart. Он предлагает высокую производительность, эффективность и безопасность при разработке приложений. Flutter обладает модульной структурой, что позволяет разработчикам создавать приложения быстро и эффективно. Он также обеспечивает множество встроенных функций, таких как виджеты для построения интерфейса, механизм управления состоянием, горячая перезагрузка для мгновенного просмотра изменений, инструменты для разработки и тестирования приложений, а также поддержку различных платформ, включая Android, iOS, веб и десктоп.

2.2. Выбор платформы для создания пользовательского интерфейса.

Виды платформ:

Веб-платформы: Веб-платформы, такие как HTML, CSS и JavaScript, позволяют создавать интерактивные и отзывчивые пользовательские интерфейсы, которые могут быть запущены в браузере. Популярные

фреймворки и библиотеки веб-разработки включают React, Angular и Vue.js. Веб-платформы обеспечивают кросс-платформенную совместимость и доступность через различные устройства и браузеры.

Мобильные платформы: Для разработки мобильных приложений можно использовать платформы, такие как iOS (с использованием языка Swift или Objective-C) и Android (с использованием языка Java или Kotlin). . Эти платформы предоставляют инструменты и фреймворки для создания пользовательского интерфейса, а также доступ к возможностям устройства, таким как камера, геолокация и датчики.

Нативные Desktop: Для создания desktop можно использовать платформы, такие как Windows (с использованием .NET Framework или C++), macOS (с использованием Objective-C или Swift) и Linux (с использованием различных технологий, таких как GTK или Qt). Нативные платформы предоставляют широкий набор инструментов для создания настольных приложений с богатым пользовательским интерфейсом.

Гибридные платформы: Гибридные платформы, такие как React Native или Flutter, позволяют создавать мобильные приложения с использованием веб-технологий. Они комбинируют преимущества веб-разработки с возможностями нативных приложений. Гибридные приложения могут быть запущены на разных plataформах, используя общий код.

В этой работе мы выберем для разработки пользовательского интерфейса веб-платформы и создадим сайт для нашего API.

2.3. Настройка среды разработки.

Для дальнейшей работы и полноценной разработки мы должны установить соответствующее для этого программное обеспечение и настроить его для себя вот несколько видов программного обеспечения для разработки на языке программирования Dart:

Android Studio: Android Studio разработана компанией JetBrains и предлагает широкий спектр инструментов для разработки приложений под Android. Она обладает мощной функциональностью, включая автодополнение

кода, отладчик, систему контроля версий, поддержку виртуальных устройств и интеграцию с другими инструментами

Visual Studio Code: Visual Studio Code (VS Code) является популярным бесплатным текстовым редактором, разработанным Microsoft. Он предлагает широкий выбор расширений и плагинов для поддержки Dart, включая автодополнение, отладчик и интеграцию с системами контроля версий.

В этой работе мы будем пользоваться средой разработки Android Studio. Так как она более удобно для разработки больших проектов.

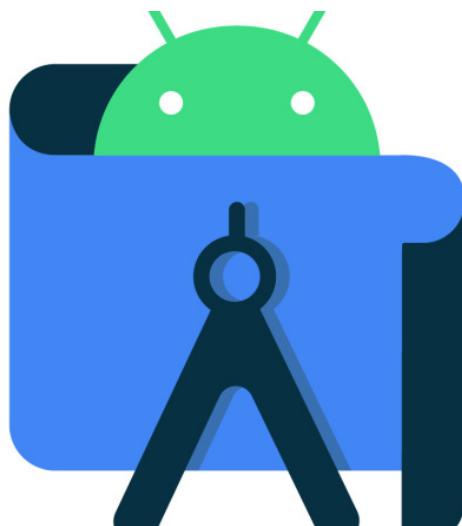


Рисунок 2.3.1 – Логотип программы Android Studio

Установка среды разработки. Скачиваем Android Studio Первый шаг — это загрузить Android Studio. Вы можете скачать его с официального веб-сайта Android Studio.

После того как вы скачали Android Studio, установите его в свою систему.

Устанавливаем Flutter SDK

Установка Flutter SDK — это второй шаг к началу разработки на Flutter.

Установка VSCode — это приложение для редактирование программы

Установка плагинов для VSCode и настраиваем под Flutter

После успешной настройки и плагинов можем приступить к разработки самого функционала и решение проблем над данной задачи.

Android Studio downloads

Download the latest version of Android Studio. For more information, see the [Android Studio release notes](#).

Platform	Android Studio package	Size	SHA-256 checksum
Windows (64-bit) Recommended	android-studio-2023.2.1.23-windows.exe	1.1 GB	754049e35fc060512eb6813e44e1a2b9dbadabe8277274fd46638e196c3285
Windows (64-bit)	android-studio-2023.2.1.23-windows-exe.zip	1.2 GB	6e0e48d2d77079e982fd791b8293edf34da1099383882783c3b77e57bbdf2917
Mac (64-bit)	android-studio-2023.2.1.23-mac.dmg	1.2 GB	b4d36685eeabc925022295ca0fe16a2ed16f45f477bc04851b34a74b347ebd5
Mac (64-bit, ARM)	android-studio-2023.2.1.23-mac_arm.dmg	1.2 GB	d5ffbfef3e0f52ef06ea15294f44000dab4ebf83fe5e30c3a3dec3a97b858e46
Linux (64-bit)	android-studio-2023.2.1.23-linux.tar.gz	1.2 GB	1b668f80ca811cdc350deb6568143da37fd57d7e9012ff5542a319e52fd5a073
ChromeOS	android-studio-2023.2.1.23-cros.deb	944.2 MB	d88ffc2ceb79e91347251b9187f14c8f802fb2fe42abcccf81d63c6f3b00f25

Рисунок 2.3.2 – Загрузка Андроид студия

Скачайте Flutter SDK

Первый шаг — загрузить Flutter SDK. Вы можете загрузить последнюю стабильную версию Flutter с официального веб-сайта Flutter

The screenshot shows the Flutter website's main landing page. At the top, there's a navigation bar with links for Multi-Platform, Development, Ecosystem, Showcase, Docs, and a search bar. Below the navigation, a banner encourages users to "Choose your development platform to get started". It features four large buttons for Windows, macOS, Linux, and ChromeOS. To the left, a sidebar titled "Get started" provides links to "Install Flutter", "Test drive", "Write your first app", "Learn more", "From another platform?", "Dart language overview", "Stay up to date", "Code labs & samples", "App solutions", "User interface", "Introduction", "Widget catalog", "Layout", and "Design & theming". A note at the bottom of the sidebar mentions using Flutter in China. A cookie consent message at the bottom states that Google uses cookies to deliver its services, personalize ads, and analyze traffic, with links to "Google settings" and "Learn more", and an "Okay" button.

Рисунок 2.3.3 – Сайт Flutter

Извлеките Flutter SDK. После загрузки Flutter SDK распакуйте его в папку в вашей системе. Например, в Windows вы можете извлечь его в папку C:\frameworks.

Устанавливаем плагины Flutter и Dart

После установки Android Studio вам необходимо установить плагины Flutter и Dart. Для этого перейдите в File -> Settings -> Plugins.

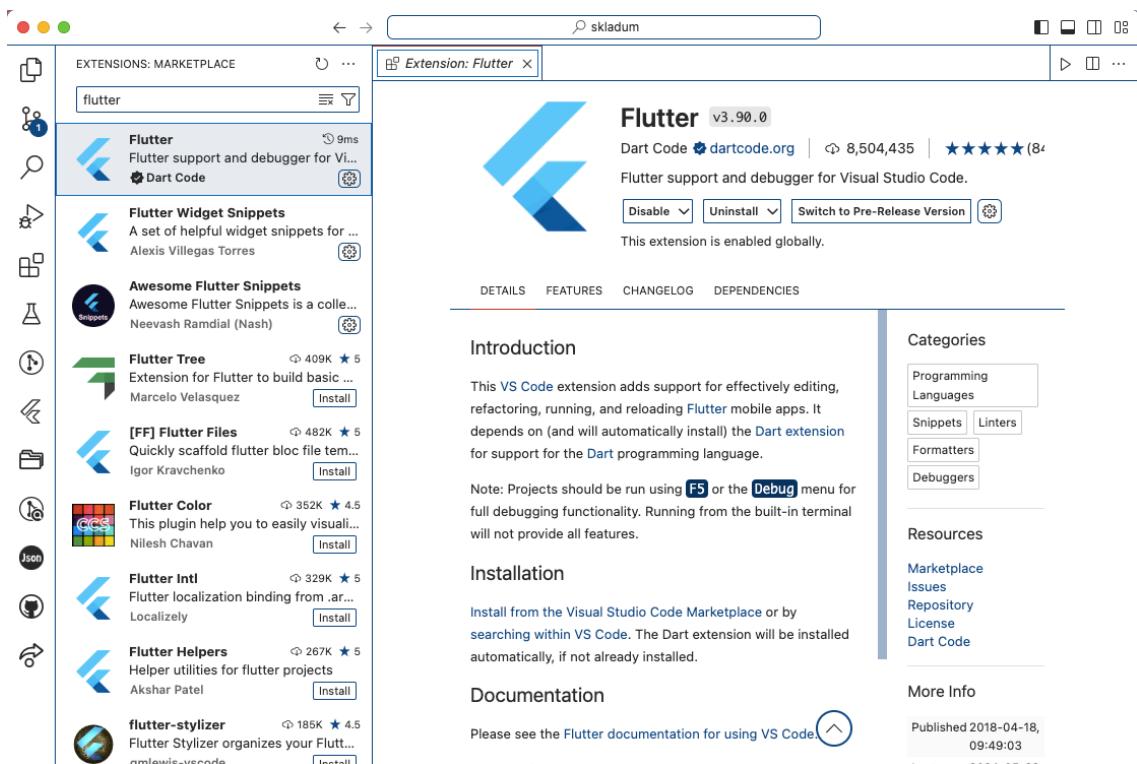


Рисунок 2.3.4 - Плагин Flutter

В окне плагинов перейдите на вкладку Marketplace и найдите «Flutter» и «Dart». Нажмите на кнопку Install, чтобы установить оба плагина.

После установке плагинов перезагрузите Android Studio

Настраиваем путь к Flutter SDK

После того как вы установили плагины Flutter и Dart, вам необходимо настроить путь к Flutter SDK. Чтобы сделать это, перейдите в File -> Settings -> Языки и фреймворки -> Flutter. В поле путь к Flutter SDK нажмите на кнопку «...» и выберите каталог, в который вы установили Flutter SDK. Нажмите на Ok, чтобы сохранить изменения.

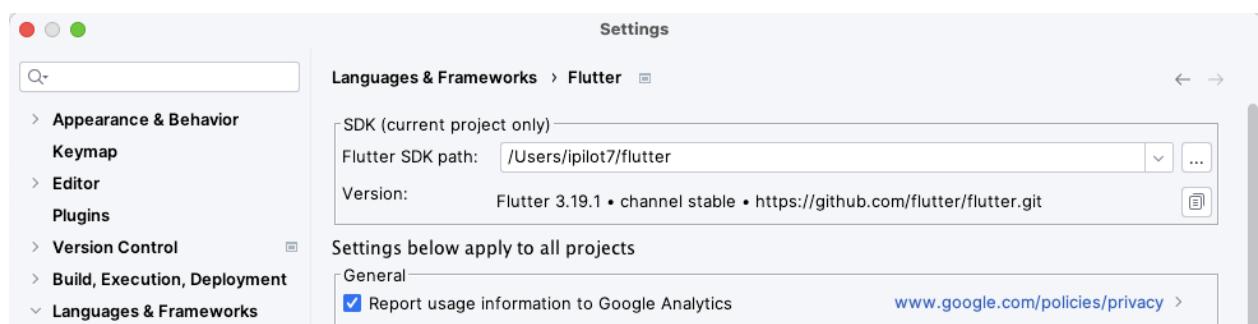


Рисунок 2.3.5 – Путь к Flutter SDK

Создаем новый проект Flutter. Теперь, когда вы настроили Flutter в Android Studio, вы можете создать новый проект Flutter. Чтобы сделать это, перейдите в File -> New -> New Flutter Project.

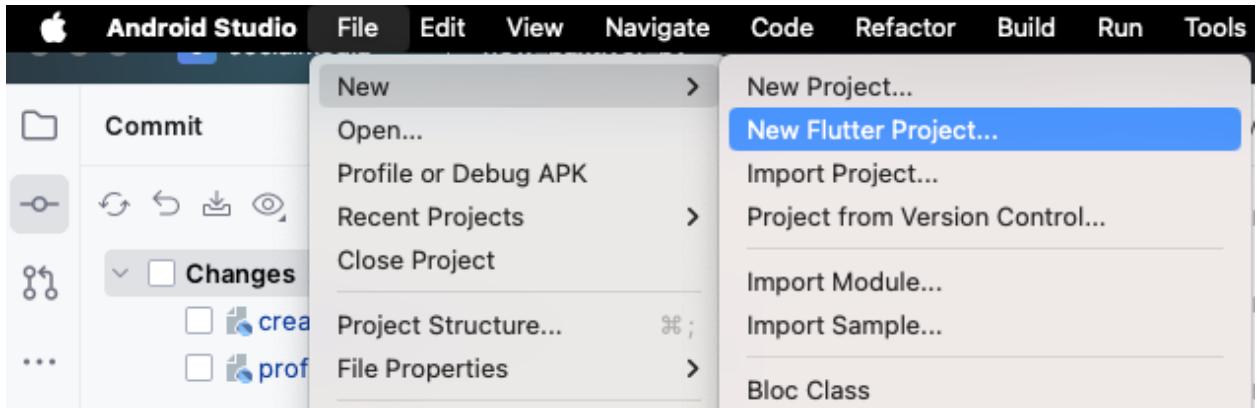


Рисунок 2.3.6 – Создание Flutter проекта

В новом окне проекта Flutter выберите вкладку Flutter и нажмите Next. В следующем окне введите название проекта, местоположение проекта и другие сведения о проекте. Нажмите на кнопку Finish , чтобы создать проект. Запустите проект Flutter

После создания проекта Flutter вы можете запустить его на эмуляторе или физическом устройстве. Чтобы запустить проект, перейдите в меню Выполнить -> Запустить ‘main.dart’. Android Studio скомпилирует проект и запустит его на выбранном устройстве.

Настройка Flutter в Android Studio — это простой и понятный процесс. Следуя инструкциям, описанным в этой статье, вы можете быстро настроить Flutter в Android Studio и приступить к разработке кроссплатформенных мобильных приложений. Flutter — это мощный фреймворк, который может помочь вам с легкостью создавать высококачественные мобильные приложения.

2.4. Создания пользовательского интерфейса.

Теперь создаём новое приложение внутри нашего проекта и называем его как frontend, создаём новое приложение следующей командой:

```
main.dart X
lib > main.dart > _MyAppState > build
Saidmirzo, 2 months ago | 2 authors (Baxtiyor and one other)
1 import 'package:easy_localization/easy_localization.dart';
2 import 'package:flutter/material.dart';
3 import 'package:flutter/services.dart';
4 import 'package:flutter_bloc/flutter_bloc.dart';
5 import 'package:skladum/common/app_color.dart';
6 import 'package:skladum/common/app_text_style.dart';
7 import 'package:skladum/common/routes.dart';
8 import 'package:skladum/di/di.dart' as sl;
9 import 'package:skladum/di/di.dart';
10 import 'package:skladum/features/history/precentration/bloc/bloc/history_bloc.dart';
11 import 'package:skladum/features/home/precentration/bloc/home_bloc.dart';
12 import 'package:skladum/features/shift/precentration/bloc/bloc/shift_bloc.dart';
13
14 Run | Debug | Profile
15 Future<void> main() async {
16   WidgetsFlutterBinding.ensureInitialized();
17
18   await sl.init();
19   SystemChrome.setSystemUIOverlayStyle(
20     const SystemUiOverlayStyle(
21       statusBarColor: Colors.black,
22       statusBarBrightness: Brightness.dark,
23       statusBarIconBrightness: Brightness.dark,
24       systemStatusBarContrastEnforced: true),
25   );
26
27   await SystemChrome.setPreferredOrientations([
28     DeviceOrientation.portraitUp,
29     DeviceOrientation.portraitDown
30   ]).then(
31     (_)>> runApp(
32       EasyLocalization(
33         useOnlyLangCode: true,
34         supportedLocales: const [
35           Locale('en'),
36           Locale('uz'),
37           Locale('ru'),
38         ],
39         path: 'assets/l10n',
40         fallbackLocale: const Locale('en'),
41         child: const MyApp(),
42       ), // EasyLocalization
43     ),
44   );
45 }
```

Рисунок 2.4.1 – main.dart файл

У нас должно было появиться директория с этим приложением и его файлами в корневой директории нашего проекта. Открываем его и создаём там новые директории с названиями templates, static и templatetags как показано на рисунке.

Все файлы показаны на рисунке 2.9. Подключение файлов в приложение – чтобы приложение работал нам нужно подключить его к нашему базовому шаблону. Делаем разметку этого меню в файле routes.dart чтобы она и в других окнах отображалась.

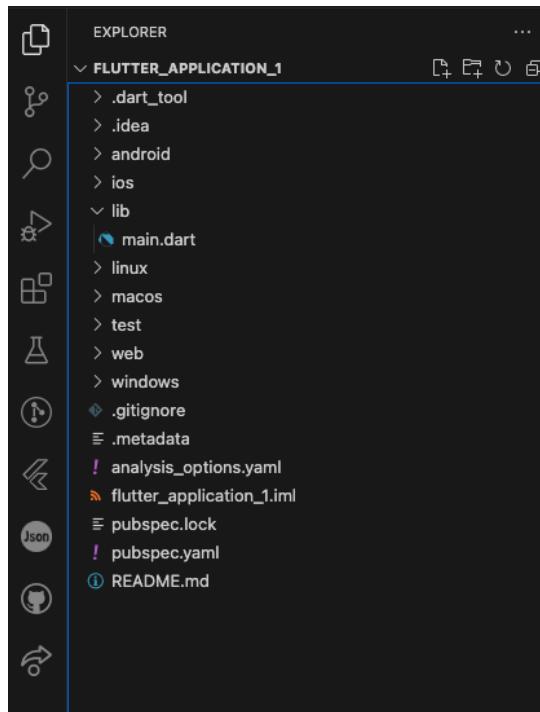


Рисунок 2.4.2 – Структура нового приложения

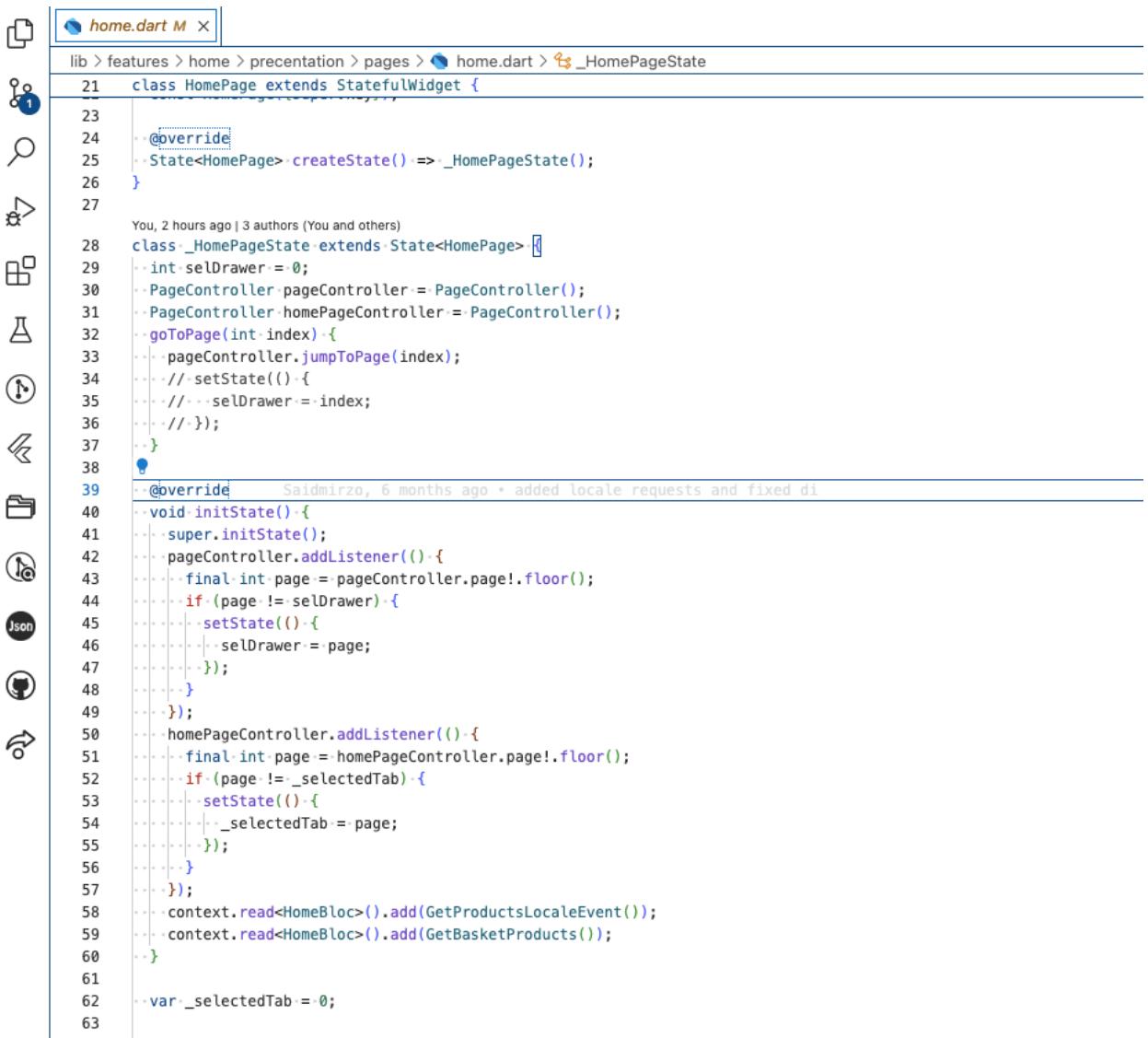
В директории `lib` создаём базовый файл и остальные файлы для нашего проекта. Эти шаблоны служат помогают нам писать код который не будет повторяться в нашем проекте много раз, таким образом шаблоны будут помогать нам соблюдать принцип проектирования DRY (Don't repeat yourself) а также с помощью этих шаблонов мы будем контролировать как будет отображаться наш приложение. Все файлы показаны на рисунке 2.9. Подключение файлов в приложение – чтобы приложение работал нам нужно подключить его к нашему базовому шаблону `main.dart` и остальные файлы добавляется и управляется через `routes.dart`.

Теперь мы можем добавлять новые файлы и связать с основным и свободно делать навигации. Теперь создадим компоненты для нашего приложение они будут включать в себя такие части:

Меню: Здесь будет отображаться вся навигация по приложению и ссылки к его основному функционалу таким как: Продажи, история, смена, настройки помошь. Делаем разметку этого меню в файле `routes.dart` чтобы она и в других окнах отображалась.

На следуем этапе мы откроем сам файл который создается при помощи Flutter CLI main.dart. Основным нашем запускающим функцией является main(). Он запускает наш проект а если не будет этого файла Flutter CLI ругается что в проекте нет функция main.

Flutter как и другие фреймворки создает собственные файлы который будет запускаться с их же помощи. Таких как: analysis_option.yaml, pubspec.yaml, pubspec.lock. Папки платформ: android, ios, web, windows, linux, macos и вместе с ним папку test для тестирование приложение и виджетов находящиеся в приложении. Писать тесты будет очень удобным и быстрым с этой папкой, так как есть готовые тесты для проверки

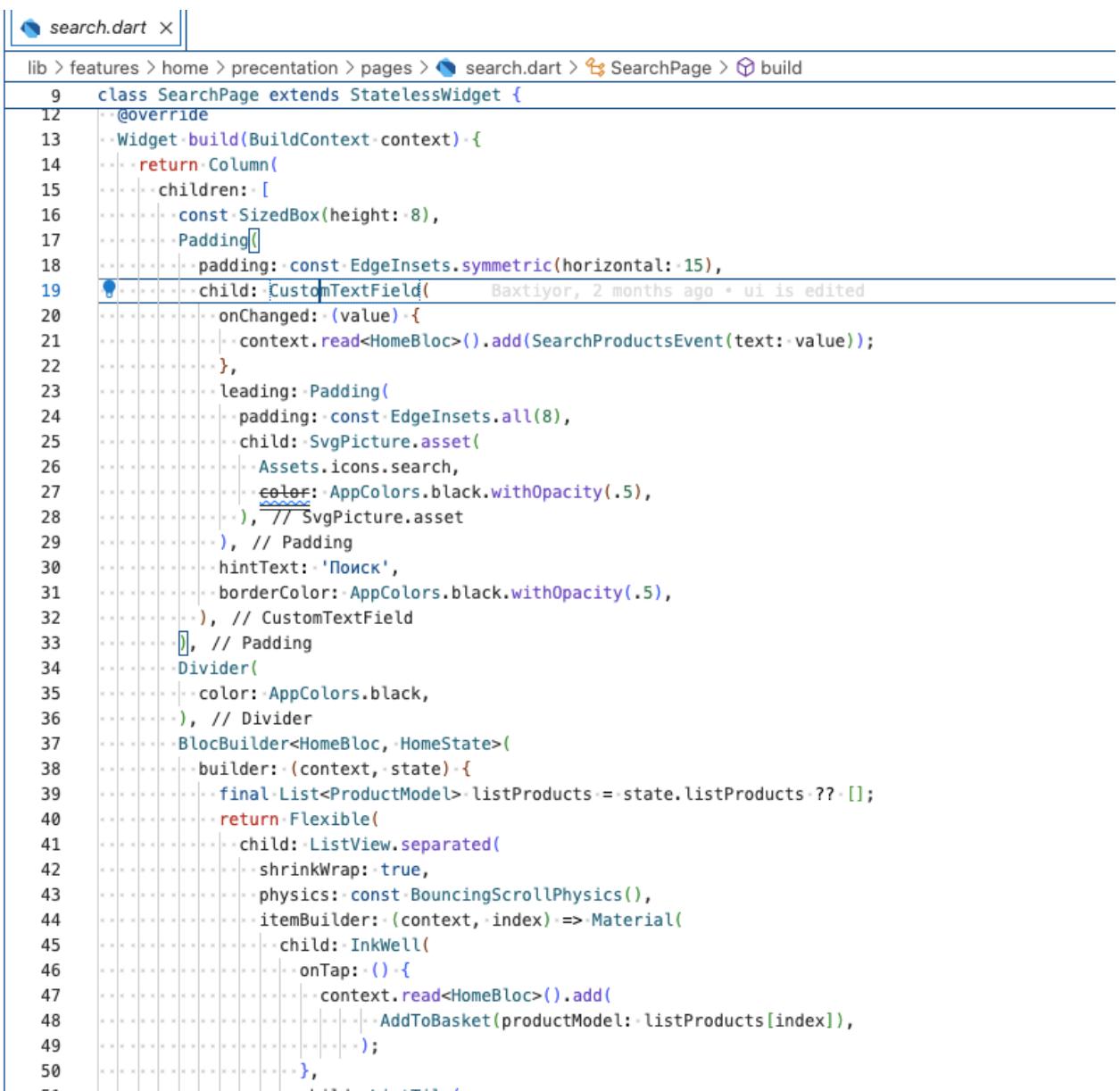


```
home.dart M X
lib > features > home > presentation > pages > home.dart > _HomePageState
21 class HomePage extends StatefulWidget {
22     ...
23
24     @override
25     State<HomePage> createState() => _HomePageState();
26 }
27
28 You, 2 hours ago | 3 authors (You and others)
29 class _HomePageState extends State<HomePage> {
30     int selDrawer = 0;
31     PageController pageController = PageController();
32     PageController homePageController = PageController();
33     goToPage(int index) {
34         pageController.jumpToPage(index);
35     }
36     setState(() {
37         selDrawer = index;
38     });
39
40     void initState() {
41         super.initState();
42         pageController.addListener(() {
43             final int page = pageController.page!.floor();
44             if (page != selDrawer) {
45                 setState(() {
46                     selDrawer = page;
47                 });
48             }
49         });
50         homePageController.addListener(() {
51             final int page = homePageController.page!.floor();
52             if (page != _selectedTab) {
53                 setState(() {
54                     _selectedTab = page;
55                 });
56             }
57         });
58         context.read<HomeBloc>().add(GetProductsLocaleEvent());
59         context.read<HomeBloc>().add(GetBasketProducts());
60     }
61
62     var _selectedTab = 0;
63 }
```

Рисунок 2.4.3 – Основная страница приложение

2.5. Сборка проекта и тестирование функциональности.

Собрать наш проект можно разными способами, и здесь мы будем использовать вариант, при котором проект будет запускаться на Андроид и iOS. Для этого мы выбрали кроссплатформенный фреймворк Flutter, который позволяет использовать единую базу кода для обеих платформ, значительно ускоряя процесс разработки и снижая затраты на поддержку. В процессе сборки мы будем использовать такие инструменты как Visual Studio Code и Xcode для настройки и тестирования приложения, обеспечивая высокую производительность и стабильность работы на различных устройствах.



```
lib > features > home > presentation > pages > search.dart > SearchPage > build
9  class SearchPage extends StatelessWidget {
12    @override
13    Widget build(BuildContext context) {
14      return Column(
15        children: [
16          const SizedBox(height: 8),
17          Padding(
18            padding: const EdgeInsets.symmetric(horizontal: 15),
19            child: CustomTextField(           Baxtiyor, 2 months ago • ui is edited
20              onChanged: (value) {
21                context.read<HomeBloc>().add(SearchProductsEvent(text: value));
22              },
23              leading: Padding(
24                padding: const EdgeInsets.all(8),
25                child: SvgPicture.asset(
26                  Assets.icons.search,
27                  color: AppColors.black.withOpacity(.5),
28                ), // SvgPicture.asset
29              ), // Padding
30              hintText: 'Поиск',
31              borderColor: AppColors.black.withOpacity(.5),
32            ), // CustomTextField
33            ), // Padding
34            Divider(
35              color: AppColors.black,
36            ), // Divider
37            BlocBuilder<HomeBloc, HomeState>(
38              builder: (context, state) {
39                final List<ProductModel> listProducts = state.listProducts ?? [];
40                return Flexible(
41                  child: ListView.separated(
42                    shrinkWrap: true,
43                    physics: const BouncingScrollPhysics(),
44                    itemBuilder: (context, index) => Material(
45                      child: InkWell(
46                        onTap: () {
47                          context.read<HomeBloc>().add(
48                            AddToBasket(productModel: listProducts[index]),
49                          );
50                        },
51                      ),
52                    ),
53                  ),
54                );
55              },
56            ),
57          ],
58        ),
59      );
60    }
61  }
```

Рисунок 2.5.1 – Пример основного файла.

Теперь нашей следующей задачей является создание файла **home.dart**, который станет центральным элементом в архитектуре нашего приложения. Этот файл будет отвечать за интеграцию и взаимодействие всех экранов приложения, предоставляя основу для единой точки входа. Файл **home.dart** не только объединит все экраны, но и будет управлять их состоянием, используя передовые практики по управлению состоянием в Flutter, такие как Provider или BLoC, что обеспечит надёжность и отзывчивость приложения.

Кроме того, в файле **routes.dart** будет реализована детализированная система маршрутизации, которая укажет порядок инициализации экранов и настройки их взаимодействий. Маршрутизация будет сконфигурирована таким образом, чтобы обеспечить оптимальное использование ресурсов устройства и мгновенный отклик на действия пользователя. Это достигается за счёт асинхронной загрузки данных и ленивой инициализации компонентов, что значительно снижает время загрузки приложения и улучшает общую производительность.

Дополнительно, мы включим в проект модуль для работы с базой данных, который будет взаимодействовать с **home.dart** для обеспечения синхронизации и актуальности данных на всех экранах. Модуль будет использовать последние достижения в области асинхронного программирования и управления состоянием, чтобы данные были консистентны и безопасно обрабатывались в многопоточной среде. Это позволит нам обеспечить стабильную и эффективную работу приложения, даже при высоких нагрузках и обширной базе пользователей.

Все выше указанные модули поддерживают данные с высокой нагрузкой в базу. Будем использовать базу данных SQLlite. Структурируя связи баз будет удобен с sql запросами и вызвать данные из базы. Мы могли бы использовать Hive но это noSql база , в этом метода все данные сохраняется в виде json формата и мы связки не сможем настроить. По этому не использовали в проекте.

```

lib > common > routes.dart > Routes > generateRoutes
19   class Routes <{
20     |   static const home = '/home';
21     |   static const login = '/login';
22     |   static const splash = '/';
23     |   static const chat = '/chat';
24     |   static const basket = '/basket';
25     |   static const comment = '/comment';
26     |   static const saleInfo = '/saleInfo';
27     |   static const payment = '/payment';
28     |   static const endPayment = '/endPayment';
29     |   static const endShift = '/endShift';
30     |   static const payMoneyPage = '/payMoneyPage';
31     |   static const paymentInCashPage = '/paymentInCashPage';
32     |   static const mixedPaymentPage = '/mixedPaymentPage';
33     |   static const discountPage = '/discountPage';
34     |   static const receiptPage = '/receiptPage';
35     |   static const returnPage = '/returnPage';
36
37   static Route<dynamic> generateRoutes(RouteSettings settings) <{
38     try {
39       final Map<String, dynamic>? args =
40         settings.arguments as Map<String, dynamic>?;
41       args ??= <String, dynamic>{};
42       switch (settings.name) {
43         case home:
44           return CupertinoPageRoute(
45             settings: const RouteSettings(),
46             builder: (_) => const HomePage(),
47           ); // CupertinoPageRoute
48         case login:
49           return CupertinoPageRoute(
50             settings: const RouteSettings(),
51             builder: (_) => LoginPage(),
52           ); // CupertinoPageRoute
53         case splash:
54           return CupertinoPageRoute(
55             settings: const RouteSettings(),
56             builder: (_) => const Splash(),
57           ); // CupertinoPageRoute
58         case chat:

```

Рисунок 2.5.2 – Пример файла навигации.

Если у вас установлен Flutter на компьютере то вводим следующую команду " flutter run " в терминале вашей операционной системы. После этого проект запустится и мы можем протестировать его работу.

```

▽ DEBUG CONSOLE
Launching lib/main.dart on iPhone 15 Pro in debug mode...
Xcode build done.                               36.2s
[ERROR:flutter/shell/platform/darwin/graphics/FlutterDarwinContextMetalImpeller.mm(42)] Using the Impeller rendering backend.
Connecting to VM Service at ws://127.0.0.1:49819/AxP66HrXY=/ws
flutter: \^[[90m[!] Easy Localization] [DEBUG] Localization initialized<...
flutter: \^[[90m[!] Easy Localization] [DEBUG] Start<...
flutter: \^[[90m[!] Easy Localization] [DEBUG] Init state<...
flutter: \^[[90m[!] Easy Localization] [DEBUG] Build<...
flutter: \^[[90m[!] Easy Localization] [DEBUG] Init Localization Delegate<...
flutter: \^[[90m[!] Easy Localization] [DEBUG] Init provider<...
flutter: \^[[90m[!] Easy Localization] [DEBUG] Load Localization Delegate<...
flutter: \^[[90m[!] Easy Localization] [DEBUG] Load asset from assets/l10n<...

```

Рисунок 2.5.3 – Запуск проекта.

ГЛАВА III. ОФОРМЛЕНИЕ РЕЗУЛЬТАТОВ И ДОСТИЖЕНИЕ ПРОЕКТА

3.1. Сборка проекта и тестирование функциональности

Мы уже создали проект, а теперь давайте будем рассматривать процесс разработки приложения.

Приложение состоит из:

Splash, Логин, Главная страница, Сканнер, Добавить товар

Splash screen. При входе в приложение открывается Splash Screen, на котором отображается логотип. Этот экран остается на экране в течение 2 секунд, создавая первое впечатление о бренде. После этого пользователь автоматически перенаправляется на страницу Логина.



Рисунок 3.1.1. Страница запуска

Логин. В этой странице пользователь с помощью логина и пароля может войти в приложение. Если он есть в базе то переходит на главную страницу.

Главная страница. В верхней части страницы расположены иконка меню который откроет дополнительные меню. Рядом с иконкой меню расположен корзинка который собирается весь заказ. В нижней части меню расположен поля быстрого и удобного поиска нужной информации о товаре. Далее представлены список товаров, позволяющие пользователю ориентироваться и легко находить интересующие его товары.

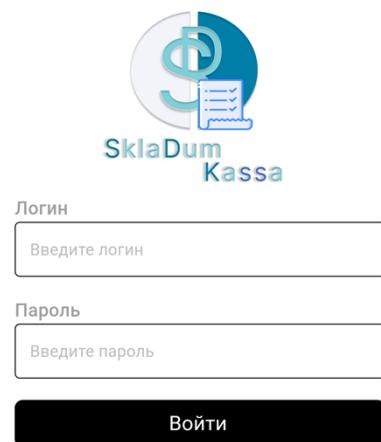


Рисунок 3.1.2. Страница Логин.

Поиска в вашем приложении позволяет пользователям находить нужные товары быстро и эффективно. Это один из ключевых компонентов, обеспечивающих удобство использования приложения, особенно когда ассортимент товаров велик.

Страница поиска разработана с целью обеспечить пользователю возможность быстро находить конкретные товары, повысить удобство и эффективность использования приложения, а также уменьшить время, затрачиваемое на поиск нужных товаров.

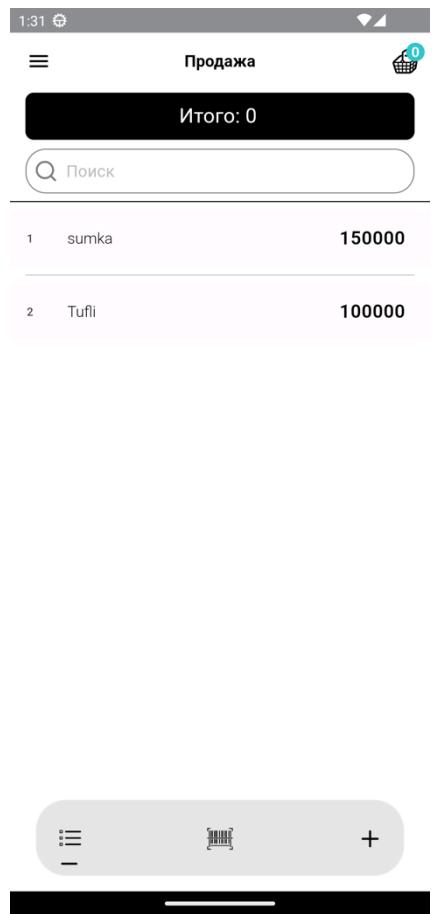


Рисунок.3.1.3. Главная страница

Её основные компоненты включают в себя поисковую строку, результаты поиска.

Сканер. Сканер поможет нам сканировать товары и автоматом добавить товар в корзину если есть в базе данных. Если нет в базе то направит в добавление товара и там же можем добавить имя товара и сохранить в базу.

Сканер очень удобен при продаже товара и быстрого нахождение и определение цен продаваемого товара. Это улучшает быстроту продаж и экономить время обслуживание. И нам не придется купить дополнительные устройства(внешний сканнер) и экономить деньги.

Добавить товар. Страница нашем приложении позволяет пользователям добавить товар. Есть два режима добавление товара. Первый нам позволяет в ручную ввести данные товара. Второе чуть облегчает сканировав штрихкод товара и з аполняет поля этим кодом нам остается ввести имя и цену на товар.

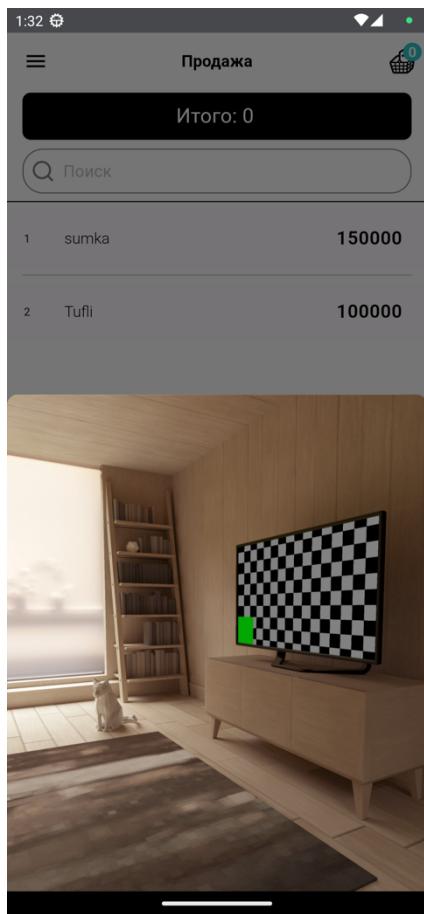


Рисунок.3.1.4. Сканер

Ручной ввод данных: Пользователи могут вручную ввести информацию о товаре, такую как наименование, цена, описание и другие атрибуты. Этот режим особенно полезен для добавления уникальных или нестандартных товаров, которые не имеют штрихкода.

Сканирование штрихкода: Для облегчения процесса добавления товара предусмотрена функция сканирования штрихкода. При сканировании штрихкода приложение автоматически заполняет поля, используя данные, связанные с этим кодом. Пользователю остается лишь ввести имя и цену на товар. Этот режим значительно ускоряет процесс добавления товаров и уменьшает вероятность ошибок, связанных с ручным вводом данных.

Экономия времени: Возможность быстрого сканирования штрихкодов и автоматического заполнения полей позволяет значительно сократить время на добавление новых товаров.

Точность данных: Автоматическое заполнение полей снижает риск ошибок, связанных с ручным вводом информации. Это особенно важно для крупных складов и магазинов с большим ассортиментом товаров.

Удобство использования: Пользователи могут выбирать наиболее удобный для них режим добавления товаров в зависимости от конкретной ситуации. Интуитивно понятный интерфейс приложения делает процесс добавления товаров простым и удобным даже для неопытных пользователей.

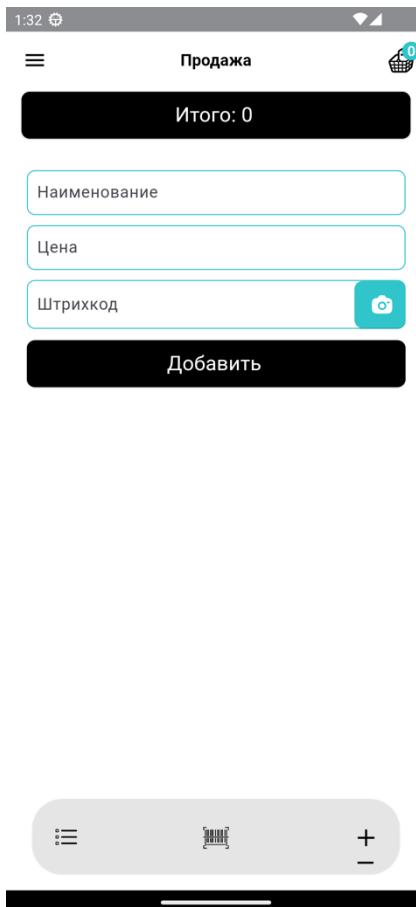


Рисунок 3.1.5. Добавить товар

Интеграция с базами данных: Приложение может быть интегрировано с различными базами данных товаров, что позволяет получать актуальную информацию о продуктах по сканированному штрихкоду. Это обеспечивает высокую точность данных и их своевременное обновление. Процесс оформления включает ввод контактной информации, выбор способа доставки и оплаты. Корзина автоматически передает всю информацию о товарах на этап оформления заказа,



Рисунок 3.1.6. Корзинка

Добавление товаров в корзину: Пользователи могут легко добавлять товары в корзину из списка продуктов или страницы с деталями товара. Для этого достаточно нажать кнопку "Добавить в корзину", и выбранный товар мгновенно появится в корзине.

Просмотр содержимого корзины: В любой момент времени пользователи могут открыть корзину и просмотреть все добавленные товары. Корзина предоставляет детальную информацию о каждом товаре, включая название, количество, цену за единицу и общую стоимость.

Редактирование корзины:

Изменение количества: Пользователи могут легко изменить количество каждого товара в корзине. Для этого предусмотрены удобные кнопки увеличения и уменьшения количества.

Удаление товаров: Если пользователь решил отказаться от какого-либо товара, он может удалить его из корзины одним нажатием кнопки "Удалить".

Общая стоимость: В корзине отображается общая стоимость всех добавленных товаров, что позволяет пользователю сразу видеть итоговую сумму заказа. Это помогает планировать бюджет и контролировать расходы.

Оформление заказа: После того, как все необходимые товары добавлены в корзину, пользователь может перейти к оформлению заказа. Процесс оформления включает ввод контактной информации, выбор способа доставки и оплаты. Корзина автоматически передает всю информацию о товарах на этап оформления заказа, что упрощает процесс и экономит время пользователя.

Сохранение корзины: Корзина автоматически сохраняет все добавленные товары, даже если пользователь закрывает приложение. Это означает, что при следующем запуске приложения корзина будет в том же состоянии, в каком она была оставлена. Это особенно удобно для пользователей, которые хотят продолжить покупки позже.

Синхронизация с учетной записью: Если пользователь зарегистрирован в приложении, корзина может синхронизироваться с его учетной записью. Это позволяет пользователю иметь доступ к своей корзине с любого устройства, просто войдя в свою учетную запись.

Удобство и простота использования: Корзина обеспечивает интуитивно понятный интерфейс для добавления и управления товарами, что делает процесс покупок более удобным и приятным.

Экономия времени: Автоматизация процесса добавления товаров и оформления заказов позволяет значительно сэкономить время пользователей.

Повышение лояльности клиентов: Удобный и быстрый процесс оформления заказов способствует повышению удовлетворенности клиентов и их лояльности к вашему бизнесу.

Улучшенное планирование покупок: Пользователи могут добавлять товары в корзину и планировать свои покупки заранее, что способствует более рациональному использованию бюджета.

Просмотр всех заказов: Пользователи могут видеть полный список всех своих предыдущих заказов, включая недавние и более старые заказы. Для удобства просмотра заказы могут быть отсортированы по дате, статусу или другим критериям.

Детали заказа: При нажатии на любой заказ из списка, пользователь получает доступ к детальной информации о данном заказе, включая:

Список товаров с указанием наименования, количества и цены каждого товара.

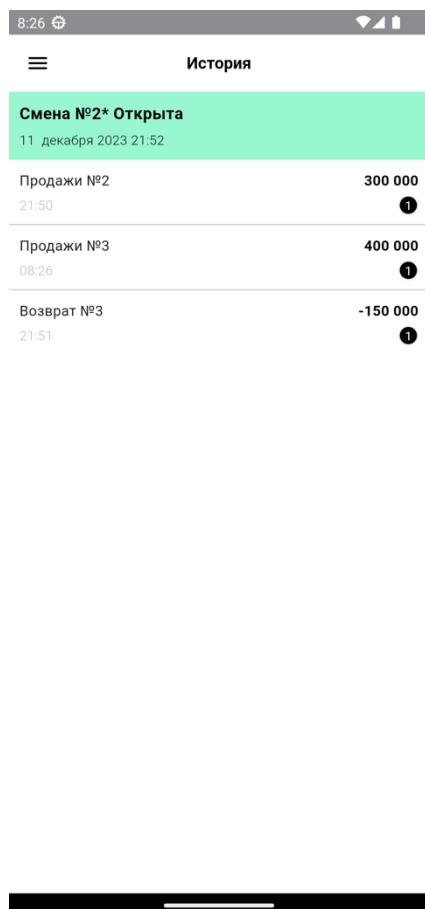


Рисунок 3.1.7. История продаж

Общая стоимость заказа.

Дата и время оформления заказа.

Статус заказа (например, "В обработке", "Отправлен", "Доставлен").

Информация о доставке, включая адрес и метод доставки.

Способ оплаты и статус оплаты.

Удобство и прозрачность: История заказов предоставляет пользователям полную информацию о всех их покупках, что способствует прозрачности и удобству использования приложения.

Анализ покупок: Пользователи могут анализировать свою покупательскую активность, что помогает лучше планировать будущие покупки и контролировать расходы.

Повышение удовлетворенности клиентов: Возможность отслеживать заказы и оставлять отзывы улучшает клиентский опыт и повышает уровень удовлетворенности пользователей.

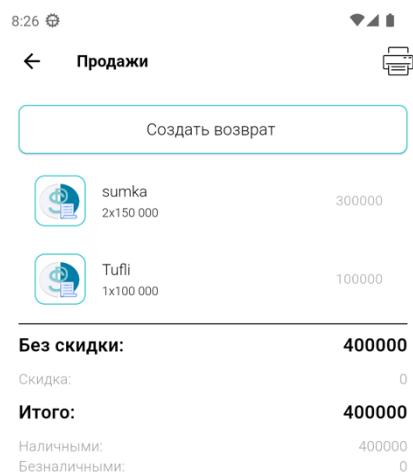


Рисунок 3.1.8. Продажа

Простота повторных покупок: Функция повторного заказа позволяет пользователям быстро и легко заказывать те же товары, что экономит их время и усилия.

Улучшение качества обслуживания: Отзывы и оценки помогают улучшить качество товаров и обслуживания, предоставляя ценную обратную связь для бизнеса.

После завершения покупки приложение автоматически генерирует электронный чек, который содержит все детали заказа. Это позволяет пользователю сразу получить подтверждение своей покупки без необходимости дополнительных действий.

Детали чека: Чек включает следующую информацию:

Название магазина или продавца.

Дата и время покупки.

Перечень приобретённых товаров с указанием наименования, количества и цены каждого товара.

Общая стоимость заказа.

Применённые скидки и акции.

Налоговые сборы (если применимо).

Способ оплаты и подтверждение оплаты.

Уникальный номер чека для идентификации.

После завершения покупки чек может быть автоматически отправлен на электронную почту пользователя. Это обеспечивает удобный доступ к чеку в любое время и позволяет сохранить его для дальнейшего использования. Все чеки сохраняются в истории заказов, что позволяет пользователю в любое время просмотреть и скачать чек для любой из своих покупок. Это удобно для ведения учета расходов и при необходимости возврата товара.

Для удобства и ускорения процессов возврата или обмена товара чек может содержать QR-код, который при сканировании предоставляет доступ ко всей информации о покупке. Это позволяет быстро идентифицировать заказ и ускоряет обработку запросов. Приложение поддерживает интеграцию с

фискальными регистраторами для автоматического создания фискальных чеков, соответствующих требованиям законодательства. Это особенно важно для соблюдения налоговых и бухгалтерских стандартов.

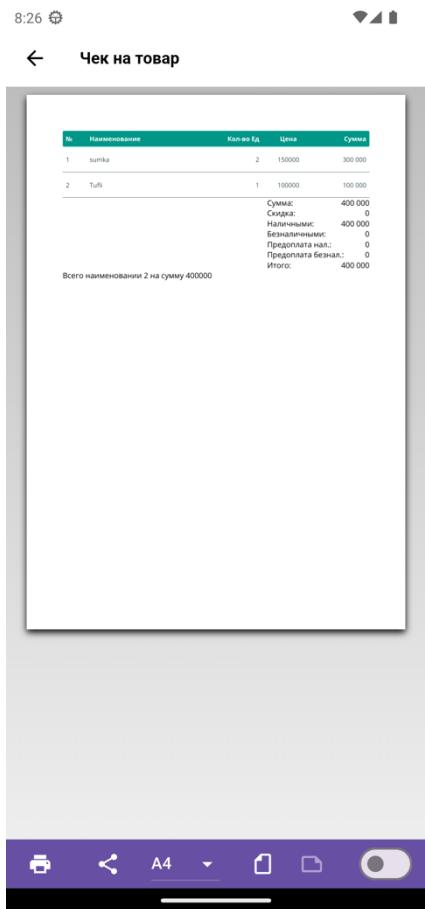


Рисунок 3.1.9. Чек на товар

Предоставление электронных чеков повышает прозрачность покупок и укрепляет доверие между продавцом и покупателем.

Электронные чеки легко сохранять и организовывать, что упрощает ведение учета расходов для пользователей. Наличие QR-кодов и детализированной информации в чеках ускоряет процесс возврата и обмена товаров, делая его более удобным для пользователей. Интеграция с фискальными регистраторами обеспечивает соответствие всех операций требованиям законодательства, что важно для юридической защиты бизнеса. Удобство получения и хранения электронных чеков улучшает общий клиентский опыт и способствует повышению лояльности покупателей.

3.2. Сохранение товаров в базу, работа с SQL базой

SQLite — это компактная встраиваемая реляционная база данных, которая не требует отдельного серверного процесса и позволяет управлять данными с высокой эффективностью и минимальными накладными расходами. Это делает SQLite идеальным выбором для мобильных приложений, веб-проектов малого и среднего размера, а также для встраиваемых систем.

Создание структуры базы данных

Для начала необходимо создать базу данных SQLite, которая будет хранить информацию о товарах. Это можно сделать с помощью следующего SQL запроса:

```
CREATE TABLE IF NOT EXISTS products (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    barcode TEXT NOT NULL,
    name TEXT NOT NULL,
    price REAL NOT NULL
);
```

В Flutter будет выглядеть так:

```
Future addToBasket(ProductModel productModel) async {
    await database.insert(basket, {"product_id": productModel.id});
} else {
    throw CacheException();
}
```

В этой таблице products:

id — уникальный идентификатор товара, который автоматически инкрементируется.

barcode — штрихкод товара, представленный в виде текста.

name — название товара.

price — цена товара.

Добавление данных в таблицу

Чтобы добавить информацию о новом товаре, можно использовать SQL запрос INSERT:

```
INSERT INTO products (barcode, name, price) VALUES  
( '1234567890123', 'Молоко', 59.99);
```

Этот запрос добавляет в таблицу products новый товар с заданным штрихкодом, именем и ценой.

Извлечение данных из таблицы

Для получения информации о товарах можно использовать запрос SELECT:

```
SELECT * FROM products;
```

В Flutter код будет выглядеть так:

```
Future<List<StudentModel>> getTeachersStudents({required  
int id}) async {  
    try {  
        if (database.isOpen) {  
            final response = await database.rawQuery(  
                '''SELECT * from student  
                WHERE id in  
                (SELECT student_id from teacher_student  
                WHERE teacher_id==$id and  
date_id==$dateId)  
                ''',  
            );  
            final listTeachers =  
List<StudentModel>.from(response.map((e) {  
                return StudentModel.fromJson(e);  
            }));  
            return listTeachers;  
        }  
    } catch (e) {  
        print(e);  
    }  
}
```

```

        }
    } catch (e) {
        log("getSpeakingViewList", error: e.toString());
    }
    return [];
}

```

Этот запрос вернет все записи из таблицы products, что позволяет просматривать всю информацию о товарах. Можно также использовать условия и фильтры для получения данных о конкретных товарах:

```
SELECT * FROM products WHERE barcode = '1234567890123';
```

Обновление и удаление данных. Для изменения информации о товаре можно использовать запрос UPDATE:

```
UPDATE products SET price = 49.99 WHERE barcode =
'1234567890123';
```

В Flutter код будет выглядеть:

```
Future<void> setPayment(StudentModel studentModel) async {
    try {
        if (database.isOpen) {
            database.rawUpdate(
                '''
                    UPDATE student
                    SET payment=${studentModel.payment},
                    payment_date=${DateTime.now().millisecondsSinceEpoch},
                    days=${studentModel.days}
                    WHERE id==${studentModel.id}''',
                );
        }
    } catch (e) {

```

```
        log("getSpeakingViewList", error: e.toString());  
    }  
}
```

Этот запрос изменит цену товара с указанным штрихкодом.

Для удаления записи из таблицы используется запрос DELETE:

```
DELETE FROM products WHERE barcode = '1234567890123';
```

В Flutter код выглядит так :

```
Future<void> deleteStudent(StudentModel studentModel) async  
{  
    try {  
        if (database.isOpen) {  
            await database.insert(  
                tableDeletedStudents,  
                studentModel.toJson(isDeleted: true),  
            );  
            database.rawQuery("DELETE FROM student WHERE  
id==${studentModel.id};");  
            deleteStudentFromTeacher(studentModel);  
        }  
    } catch (e) {  
        log("getSpeakingViewList", error: e.toString());  
    }  
}
```

Этот запрос удаляет товар с заданным штрихкодом из базы данных.

ГЛАВА IV. ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ

4.1 Работа с компьютерным оборудованием и анализ факторов риска

Охраной труда называют систему законодательных актов и соответствующих им социально-экономических, технических, гигиенических и организационных мероприятий, обеспечивающих безопасность, сохранение здоровья и работоспособности человека в процессе труда. Отдельными разделами охраны труда в нашей стране являются техника безопасности, производственная санитария и пожарная защита.

Законодательство охране труда охватывает законодательные мероприятия, обеспечивающие рабочим и служащим здоровье и безопасные условия работы, и организацию надзора за выполнением законов и правил по охране труда.

Техника безопасности представляет собой систему организационных и технических мероприятий и средств, предотвращающих воздействие на работающих вредных производственных факторов. Методы и технические средства, с помощью которых осуществляют профилактику производственного травматизма, являются основным содержанием техники безопасности. В производственной санитарии называют систему организационных, гигиенических и санитарно-технических мероприятий и средств, предотвращающих воздействие на работающих вредных производственных факторов.

Пожарная защита - это комплекс организационных и технических средств, направленных на предотвращение воздействия на людей опасных факторов пожара и ограничение материального ущерба от него.

Развитие производства на основе комплексной механизации и широкой автоматизации производственных процессов открывает в нашей стране широкие возможности для дальнейшего оздоровления и облегчения труда и ликвидации причин, порождающих производственный травматизм и профессиональные заболевания

Инженерно-технические работники и администрация организации несут административную и уголовную ответственность за нарушение возложенных на них обязанностей по соблюдению правил и норм охраны труда.

Административную или дисциплинарную ответственность влекут за собой нарушения, которые не носят злостного характера и не привели к тяжелым последствиям.

Нарушение правил, в результате которого произошла авария или несчастный случай, влечет за собой уголовную ответственность. Как преступное может рассматриваться и поведение лица, упорно не желающего устраниить нарушение правил охраны труда, тем более если такое нарушение повлекло или могло повлечь тяжелые последствия.

Нарушение должностным лицом правил охраны труда, если это нарушение могло повлечь за собой несчастные случаи с людьми или иные тяжкие последствия, наказывается лишением свободы на срок до одного года или исправительными работами на тот же срок, штрафом или увольнением с занимаемой должности. Если же нарушения повлекли за собой причинение телесных повреждений или утрату трудоспособности, то виновный несет наказание в виде лишения свободы на срок до трех лет или исправительных работ на срок до одного года. Нарушения, повлекшие смерть человека или причинение тяжких телесных повреждений нескольким лицам, наказываются лишением свободы на срок до пяти лет.

Успешная борьба с производственным травматизмом немыслима без тщательной проверки обстоятельств нарушения правил охраны труда. Закон четко определил круг лиц - субъектов преступления, которые могут нести ответственность за нарушение этих правил.

Субъектом преступления является не всякое должностное лицо, а лишь то, на которое возложена ответственность за обеспечение безопасных условий труда. Поэтому при расследовании обстоятельств, связанных с несчастным случаем на производстве, нужно тщательно проверить, на кого из

должностных лиц возложена обязанность обеспечивать соблюдение правил охраны труда и техники безопасности.

Вышесказанное, конечно, не означает, что рядовые исполнители работ не несут никакой ответственности, если по их вине на производстве произошел несчастный случай. Может наступить ответственность в уголовном порядке и рядового исполнителя работ, повинного в нарушении общих правил охраны труда и техники безопасности, если следствием этого явилось причинение вреда здоровью окружающих.

Дисциплина труда, чувство ответственности за порученное дело обязывают не только администрацию, но и всех сотрудников строго соблюдать правила техники безопасности и промышленной санитарии.

Грубая неосторожность потерпевшего, равно как и пренебрежение им правилами техники безопасности, может исключить ответственность администрации за несчастный случай, но только при условии, если потерпевший действовал вопреки прямому запрету администрации, со стороны которой нарушений допущено не было.

Искоренение нарушений правил охраны труда и техники безопасности заключается не только в уголовно-правовой борьбе с этими нарушениями, но и в их предупреждении, профилактике. Одними из эффективных профилактических мероприятий по охране труда являются организация обучения и проверки знаний правил охраны труда руководящих инженерно-технических работников и инструктаж и обучение рабочих безопасным методам труда.

Прежде чем приступить к организации места, в дипломной работе следует показать, для каких целей разрабатываются программы и где они используются, производство, (банковская система). Это обеспечивает хорошую оценку всех рисков, возникающих при работе с компьютером. После этого решаются вопросы, связанные с компьютерными классами и их размещением, определением площади для одного компьютера, обеспечением рабочего места инвентарем.

(стол, стул, шкаф и т.п.) рассматриваются вопросы, кроме того, анализируются опасные и вредные факторы, возникающие при работе компьютера. В этом разделе рассматриваются следующие вопросы:

- необходимая площадь помещения и требования к ней;
- требования к столам и стульям (размер, высота);
- учет антропометрических размеров при работе на компьютере;
оптимальное расстояние до глаз на экране монитора;
Причины устроиться на работу за компьютером.

Требования для работы на компьютере

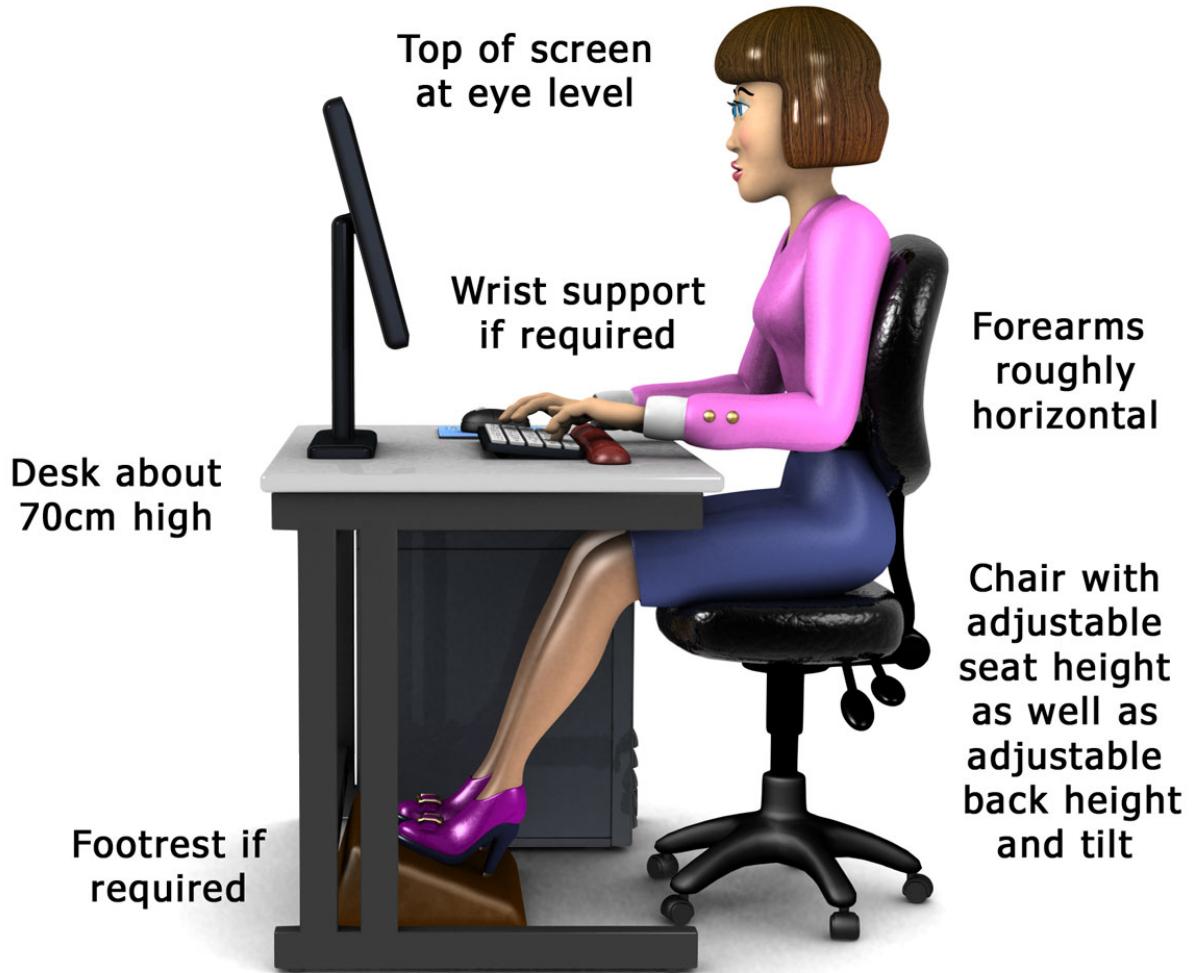


Рисунок 4.1.1. Использование компьютера.

Требования к компьютерным залам. Прежде чем приступить к организации рабочего места, в дипломной работе следует указать, для каких целей разрабатываются программы и где они используются, производство

(банковская система). Это обеспечивает хорошую оценку всех рисков, возникающих при работе с компьютером. После этого анализируются вопросы, связанные с компьютерными классами и их размещением, определением площади для одного компьютера, обеспечением рабочего места инвентарем (стол, стул, шкаф и т.п.). Опасные и вредные факторы. В этом разделе рассматриваются следующие вопросы:

Климатические условия в помещении, где проводятся работы, должны быть достаточными; нам нужно проверять параметры микроклимата и знать, что они безвредны; состояние воздушной среды помещения должно быть свежим воздухом; что качество воздуха безвредно для здоровья человека; в рабочей зоне не должно быть пыли и других загрязняющих веществ, а плохой воздух должен быть удален от них через кондиционер; должен иметь защиту от электромагнитного поля и рентгеновского излучения; к воздухообмену в помещениях необходимо отнестись серьезно; Должна быть естественная и искусственная вентиляция.

4.2. Пожарная безопасность

Помещение, где проводятся работы, должно быть естественно освещено; Необходимо измерить, является ли влияние солнечной радиации на микроклимат помещения малым или большим: цветовая отделка помещений должна быть безвредна для глаз людей, помещения должны быть изолированы от солнечного излучения, искусственного освещения. система должна быть достаточной, должна работать на уровне; при выборе осветительных приборов следует выбирать осветительные приборы, безвредные для человека; уровень освещения также должен в меру освещать помещение.

- производственный шум в компьютерных залах должен быть умеренным и работать следует в местах, защищенных от чрезмерного шума;
- утепление ограждающих конструкций и внутренних стен помещения должно быть непрерывно функциональным.

Электрическая безопасность

- мы должны работать в местах, где нет риска поражения электрическим током;
- необходимо заземлить компьютерное оборудование и обеспечить передачу избыточного напряжения на землю;
- мы должны работать на линиях электропередачи и использовать средства общественной и индивидуальной защиты;
- первая помощь при ударе током должна быть выставлены на показ аптечки для оказания медицинской помощи.

Пожарная безопасность

- причины пожара необходимо изучить и устранить;
 - передающее оборудование должно быть всегда наготове;
 - пути эвакуации в случае пожара следует изучать и каждый должен об этом знать;
 - потушить огонь водой нам нужно обеспечить отключение.
 - климат в помещении;
- параметры микроклимата и их описание;
- кондиционер в номере;
- качественный состав воздуха;
- наличие пыли и других загрязняющих веществ в рабочей зоне;
- электромагнитное поле и рентгеновское излучение;
- воздухообмен в помещениях;
- естественная вентиляция;
 - искусственная вентиляция;
 - Конденсация воздуха.

Пожарная безопасность

Температура и влажность воздуха, скорость движения воздуха, давление и тепло, выделяемое оборудованием и выпускаемой продукцией. Указанные факторы рассчитываются на производственный микроклимат данного цеха. Необходимо учитывать физические возможности и трудовые особенности лиц, допущенных к работе на производстве. Обслуживающие работники

должны пройти профессиональную подготовку по охране труда в соответствии с выполняемой ими работой.

Специальные инструменты – используются для выполнения специальных задач при тушении пожаров. К ним относятся грузовые автомобили и краны, средства освещения и связи, а также оперативные транспортные средства.

Вспомогательные средства – создают достаточные условия для проведения работ по тушению пожара. К ним относятся водонагреватели, грузовики, автобусы, тракторы и другие транспортные средства.

Радиоактивные лучи также дают хорошие результаты при анализе состава кристаллических веществ, контроле и автоматизации производственных процессов. Ионизированные лучи губительно действуют на организм человека и могут вызвать тяжелые заболевания. Под его влиянием человек может страдать от легких, белой болезни крови, а также от различных опасных опухолей и заболеваний кожи, которые являются серьезными заболеваниями. Кроме того, ионизированные лучи могут вызывать генетические эффекты, то есть наследственные заболевания, поражающие следующие поколения. защитных мер обеспечивают безопасность.



Рисунок 4.2.1. Пожарная безопасность.

Начинаем разрабатывать мероприятия по каждому пунктуанализируются ранее существовавшие опасные и вредные факторы. Мероприятия, которые выпускник хочет использовать на практике, ясны, предусмотрены санитарно-гигиенические, технические и организационные мероприятия, которые должны предотвратить влияние негативных факторов и создать благоприятные условия для работы.

Завершается раздел подборкой инструкций по компьютерной безопасности.

Назначение. Расчет общей освещенности пола шириной А м и (высотой) В м люминесцентными лампами: высота белого потолка Нм Рп = 70%, стены светлые, окна незамаскированные Rs=50%. С низким выбросом пыли и дыма. Стандартно требуется свет Эм лк. Светильник представляет собой решетчатый светильник прямого рассеяния света (15°) с люминесцентной лампой ДС-30, световой поток F3=1160 лм. Мы принимаем.

Метод расчета.

$$F_l = \frac{E_m \cdot K \cdot S \cdot Z}{N \cdot \eta} = 200 * 1,5 * 432 * 0,95 / 8,67 * 38 = 373,7$$

Здесь Fl – световой поток каждой лампы, лм;

Em – норма освещенности, лк;

K – коэффициент резерва

S – площадь помещения, м²

N – количество ламп

η - коэффициент использования светового потока, то есть общий световой поток во всех помещениях в зависимости от токов, падающих на рабочую поверхность.

Z- отношение средней освещенности к минимальной освещенности

$$Z = 1,15 \div 1,2$$

1. Определение единицы измерения помещения.

$$I = \frac{S}{H(A+B)} = 432 / 6 * (18 + 24) = 1,71$$

Здесь: S – площадь помещения, 432м².

A - ширина помещения, 18м.

V - длина помещения, 24м.

H - расчет высоты (расстояние от рабочего места до освещения), 6м

2. Определение высоты расчетным путем.

$$h = N - hc - hp = 6 - 0,5 - 0,8 = 4,7$$

Здесь: N - высота помещения, 6м.

hc — высота светильника, hc = 0,5м.

hp - высота рабочего места hp = 0,8 м.

3. Чтобы определить количество светильников, сначала нужно найти расстояние между ними – L. Самое удобное соотношение – когда светильники расположены в несколько рядов.

$$L: h = 1,5; \Lambda = 1,5h = 1,5 * 4,7 = 7,05$$

4. Определение количества светильников по ширине и длине помещения:

$$N_{(A)} = \frac{A}{L} = 18 / 7,05 = 2,55 \\ N_{(B)} = \frac{B}{L} = 24 / 7,05 = 3,4$$

5. Общее количество светильников определяется следующим образом:

$$H = H(A)H(B) = 2,55 * 3,4 = 8,67$$

6. Определить световой поток каждой лампы.

Если каждая лампа Фл превышает расчетную отметку светового потока (Φ_3), количество ламп следует пересчитать.

Начальные данные. К - резервный коэффициент

Таблица 4.2.1

Описание номера	Резервные коэффициенты		
	Флюоресцентные лампы	Лампы накаливания	Пора почистить фары
Большой выход пыли, дыма, тел	2	1,7	4 раза в 1 месяц

Средняя мощность пыли, дыма, учреждений	1,8	1,5	3 раза в 1 месяц
Меньше выбросов пыли, дыма, тел	1,5	1.3	2 раза в 1 месяц
В открытом состоянии	1,5	1.3	3 раза в 1 год

Коэффициент использования разных типов люминесцентных ламп

Таблица 4.2.2

Индекс площади помещения	КПД ламп											
	Лампа представляет собой сетчатый абажур с прямым рассеянным светом под углом 15°.				Жалюзи представляет собой открытый подвесной светильник наверху.				Плафоны 30° с решетчатым затемнением			
	30	50	70	50	50	70	70	50	50	50	70	
	10	30	50	30	50	50	70	30	50	50	70	
0,5	15	17	20	13	15	17	23	13	15	15	19	
0,6	19	22	24	16	19	22	17	16	19	19	22	
0,7	23	25	27	19	21	24	30	19	21	21	24	
0,8	25	27	30	21	23	26	32	21	22	23	25	
0,9	27	29	31	22	24	28	33	22	23	24	26	
1,0	29	30	32	23	25	30	35	23	24	24	27	
1,1	30	31	33	24	27	28	36	24	25	25	28	
1,25	31	33	35	26	28	30	38	25	26	26	29	
1,5	33	34	37	28	30	31	40	26	27	28	30	
1,75	34	36	38	29	32	33	41	27	28	29	31	
2,0	36	37	39	31	33	35	43	28	29	30	32	
2,25	37	38	41	32	35	37	44	29	30	31	33	

2,5	38	39	<u>42</u>	33	36	39	46	30	31	32	33	33
3.0	39	40	<u>43</u>	35	37	40	47	31	32	33	33	34
3,5	40	41	<u>44</u>	36	38	46	49	32	32	33	33	34
4.0	41	42	<u>45</u>	36	39	47	50	32	33	34	34	35
5.0	42	43	<u>46</u>	39	41	49	52	33	34	34	34	35

Варианты решения задачи.

Таблица 4.2.3

Параметры		1	2	3	4	5	6	7	8	9	10	11	12	13	14	$\frac{1}{5}$
\mathcal{E}_m , лк		50 0	30 0	40 0	20 0	30 0	50 0	20 0	40 0	30 0	20 0	50 0	40 0	30 0	50 0	2 0 0
Размеры помешения	Высота, м	15	25	70	50	70	25	24	30	24	18	10	12	14	20	3 0
	Ширина, м	10	15	50	30	35	15	18	18	12	12	8	10	11	25	2 2
	Высота, м	5	6	7	6	7	5	6	7	6	9	4	5	3	4	5 6
Параметры		16	17	18	19	20	21	22	23	24	25	26	27	28	29	3 0
\mathcal{E}_m , лк		30 0	40 0	20 0	50 0	30 0	50 0	20 0	40 0	30 0	20 0	50 0	40 0	30 0	50 0	2 0 0
Размеры помешения	Высота, м	28	23	40	35	45	25	24	30	24	18	10	12	14	20	3 0
	Ширина, м	20	10	30	28	40	15	18	18	12	12	8	10	11	25	2 2
	Высота, м	7	5	4	7	8	5	6	7	6	9	4	5	3	4	5 6

ЗАКЛЮЧЕНИЕ

В данной дипломной работе был проведен исследовательский анализ и разработка онлайн сервиса автоматизации продаж в магазинах. Целью работы было создание эффективного инструмента, позволяющего пользователям управлять и продать свои товары с помощью автоматизации процесса.

В результате данной работы был разработан сервис автоматизации продаж в магазинах, который предоставляет пользователям удобный инструмент для продаж и управления их товаров. Применение технологий Flutter и Fast API позволило создать надежный и функциональный сервис, обладающий гибкостью и возможностью расширения функциональности в будущем. Разработанный сервис имеет потенциал для применения в различных сферах деятельности, где требуется автоматизация продаж и управления их товаров. Он может быть использован как инструмент для супермаркетов минимаркетов, оптовых магазинах, и других пользователей, стремящихся оптимизировать свою работу в продаже. Результаты работы могут служить основой для дальнейших исследований и улучшений в области автоматизации продаж и разработки онлайн-сервисов. Все использованные в работе технологии, предоставили надежную основу для создания качественного и эффективного сервиса. Благодаря этим технологиям удалось реализовать требуемые функции и обеспечить удобство использования для конечных пользователей. Данная дипломная работа посвящена актуальной теме автоматизации продаж и управления. Она представляет собой полезный вклад в развитие продуктивности и может в дальнейшем выйти на рынок этой области.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

Основные литературы

1. Anderson, B., & Lan, E. (2018). *Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart*. 2. Packt Publishing:
<https://www.packtpub.com/product/flutter-for-beginners/9781788996082>.
2. Napoli, M. (2019). *Beginning Flutter: A Hands-On Guide to App Development*. Apress. <https://link.springer.com/book/10.1007/978-1-4842-4320-2>
3. Spindler, M. (2020). *Flutter Projects: A practical, project-based guide to building real-world cross-platform mobile applications and games*. Packt Publishing. <https://www.packtpub.com/product/flutter-projects/9781838647770>
4. Lewis, J., & West, R. (2019). *Flutter in Action*. Manning Publications. <https://www.manning.com/books/flutter-in-action>
5. Balbaert, I. (2022). *Flutter Complete Reference: Create beautiful, fast and native apps for any device*. Packt Publishing.
<https://www.packtpub.com/product/flutter-complete-reference/9781801817764>
6. Collins, K. (2020). *Flutter for Dummies. For Dummies*. <https://www.dummies.com/book/technology/programming/mobile-apps/flutter-for-dummies-282776/>
7. Max, D. (2019). *Flutter for Absolute Beginners: Learn to Develop Cross-Platform Apps*. Apress. <https://link.springer.com/book/10.1007/978-1-4842-4414-8>
8. Raichura, P. (2020). *Mastering Flutter: A Step-by-Step Guide to Build High-Performance Apps for iOS and Android*. Packt Publishing. <https://www.packtpub.com/product/mastering-flutter/9781838647770>
9. Nelson, B. (2020). *Flutter Recipes: Mobile Development Solutions for iOS and Android*. Apress. Provides practical solutions to common mobile development problems using Flutter.
10. Sells, C., & Shepard, A. (2021). *Flutter for the Impatient: Building Apps with Widgets*. Pearson Education. This book simplifies Flutter app development with a focus on widget use.

11. Singleton, T. (2021). Pro Flutter for Designers. Apress. Explores how designers can leverage Flutter for high-fidelity UI/UX design without deep diving into complex coding.
12. Nelson, B. (2020). Flutter Recipes: Mobile Development Solutions for iOS and Android. Apress. Provides practical solutions to common mobile development problems using Flutter.
13. Windmill, E. (2020). Flutter in Focus: Learn Flutter features in 20 Minutes. Manning Publications. Offers quick, focused lessons on key aspects of Flutter.
14. Greenhalgh, D. (2021). Advanced Flutter: Beyond Basics. Packt Publishing. Delves into more complex Flutter topics like advanced animations and custom plugin development.
15. Pilgrim, M. (2022). Efficient Flutter: Best Practices for Performance. O'Reilly Media. Focuses on optimizing Flutter applications for better performance and efficiency.
16. Dyck, J. (2021). Flutter for Web Developers. Apress. This book explores how to build web applications using Flutter, targeting web technologies specifically.
17. Kehe, G. (2020). Flutter Cookbook: Over 100 Techniques and Solutions for App Development with Flutter. O'Reilly Media. A comprehensive collection of recipes for solving specific app development problems.

Список интернет ресурсов.

1. flutter.dev (flutter.io) [En]
2. dart.dev [En]
3. Flutter for web [En]
4. Flutter for desktop [En]
5. Flutter Codelabs [En]
6. Flutter Community (Medium) [En]
7. Flutter Tutorials Handbook [En]
8. Flutter by Google [En]

ПРИЛОЖЕНИЕ

Файл зависимостей: pubspec.yaml

```
name: skladum
description: A new Flutter project.
publish_to: 'none' # Remove this line if you wish to publish
to pub.dev
version: 1.0.0+1
environment:
  sdk: '>=3.1.5 <4.0.0'
dependencies:
  auto_size_text: ^3.0.0
  barcode_scan2: ^4.3.0
  cupertino_icons: ^1.0.2
  dartz: ^0.10.1
  device_info_plus: ^9.1.1
  dio: ^5.3.3
  easy_localization: ^3.0.3
  equatable: ^2.0.5
  flutter:
    sdk: flutter
  flutter_bloc: ^8.1.3
  flutter_local_notifications: ^16.1.0
  flutter_localization: ^0.1.14
  flutter_svg: ^2.0.9
  freezed: ^2.4.3
  freezed_annotation: ^2.4.1
  get_it: ^7.6.4
  internet_connection_checker: ^1.0.0+1
  intl: any
```

```
lottie: ^2.7.0
mask_text_input_formatter: ^2.5.0
mobile_scanner: ^3.5.5
printing: ^5.11.1
shared_preferences: ^2.2.1
shimmer: ^3.0.0
sqflite: ^2.3.0
url_launcher: ^6.2.2
dev_dependencies:
  build_runner: ^2.4.7
  flutter_lints: ^2.0.0
  flutter_launcher_icons: "^0.13.1"
  flutter_test:
    sdk: flutter
flutter_launcher_icons:
  android: "launcher_icon"
  ios: true
  image_path: "assets/images/kassa_icon.png"
  min_sdk_android: 21 # android min sdk min:16, default 21
flutter:
  uses-material-design: true
assets:
  - assets/images/
  - assets/icons/
  - assets/l10n/
  - assets/lottie/
  - assets/db/

```

lib/features/home/precentration/pages/home.dart

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

```
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_svg/svg.dart';
import 'package:skladum/common/app_color.dart';
import 'package:skladum/common/app_text_style.dart';
import 'package:skladum/common/assets.dart';
import
'package:skladum/common/components/gradient_button.dart';
import
'package:skladum/common/components/univer_dialog.dart';
import 'package:skladum/common/constants.dart';
import 'package:skladum/common/enums/bloc_status.dart';
import 'package:skladum/common/routes.dart';
import 'package:skladum/features/help/help.dart';
import
'package:skladum/features/history/precentration/pages/history
.dart';
import
'package:skladum/features/home/data/model/product_model.dart
';
import
'package:skladum/features/home/precentration/bloc/home_bloc.d
art';
import
'package:skladum/features/home/precentration/pages/add.dart';
import
'package:skladum/features/home/precentration/pages/products.d
art';
```

```
import
'package:skladum/features/home/precentration/pages/search.dar
t';
import
'package:skladum/features/home/precentration/widgets/scaner_w
idget.dart';
import
'package:skladum/features/login/presentation/widgets/tabbar.
dart';
import 'package:skladum/features/settings.dart';
import
'package:skladum/features/shift/precentration/pages/shift_pag
e.dart';
class HomePage extends StatefulWidget {
  const HomePage({super.key});
  @override
  State<HomePage> createState() => _HomePageState();
}
class _HomePageState extends State<HomePage> {
  int selDrawer = 0;
  PageController pageController = PageController();
  goToPage(int index) {
    pageController.jumpToPage(index);
    setState(() {
      selDrawer = index;
    });
  }
  @override
  void initState() {
```

```
super.initState();

context.read<HomeBloc>().add(GetProductsLocaleEvent());
context.read<HomeBloc>().add(GetBasketProducts());
}

@Override
Widget build(BuildContext context) {
    return DefaultTabController(
        initialIndex: 0,
        length: 4,
        child: Scaffold(
            resizeToAvoidBottomInset: true,
            floatingActionButton: FloatingActionButton(
                child: const
Icon(Icons.cleaning_services_outlined),
                onPressed: () {
                    context.read<HomeBloc>().add(ClearDbEvent());
                },
            ),
            appBar: AppBar(
                centerTitle: true,
                // elevation: 0,
                title: Text(appbarTitles[selDrawer]),
                actions: [
                    selDrawer == 0
                        ? IconButton(
                            onPressed: () {
                                Navigator.pushNamed(context,
Routes.basket);
```

```

        },
        icon: SvgPicture.asset(
            Assets.icons.basket,
            width: 25,
        ),
    )
: const SizedBox.shrink(),
selDrawer == 1
? IconButton(
    onPressed: () {},
    icon: SvgPicture.asset(
        Assets.icons.search,
        width: 25,
    ),
)
: const SizedBox.shrink(),
],
bottom: selDrawer == 0
? PreferredSize(
    preferredSize: const
Size(double.infinity, 90),
        child: BlocConsumer<HomeBloc, HomeState>(
            listener: (context, state) {
                if (state.getBasketProductsStatus ==
                    BlocStatus.completed) {}
                if (state.createSaleStatus ==
                    BlocStatus.completed) {}
            },
            builder: (context, state) {

```

```
final List<ProductModel> list =
    state.listBasketProducts ?? [];
int allPrice = 0;
for (var element in list) {
    allPrice +=
        (element.price ?? 0) *
(element.count ?? 1);
}
return Column(
    children: [
Padding(
    padding:
        const
EdgeInsets.symmetric(horizontal: 15),
    child: GradientButton(
        color: AppColors.black,
        onPressed: () {
            if (allPrice != 0) {
                Navigator.pushNamed(
                    context,
                    Routes.payment,
                    arguments: {"allPrice": allPrice},
                );
            }
        },
        text: 'Итого: \$allPrice',
    ),
),

```

```
        const TabBarWidget()
    ],
);
},
),
)
: null),
body: PageView(
    controller: pageController,
    physics: const NeverScrollableScrollPhysics(),
    children: [
        TabBarView(
            children: [
                const ProductsPage(),
                const ScanerWidget(),
                SearchPage(),
                AddPage(goHome: () {
                    goToPage(0);
                })
            ],
        ),
        HistoryPage(),
        ShiftPage(),
        SettingsPage(),
        HelpPage()
    ],
),
drawer: Drawer(
    child: SafeArea(
```

```
        child: Column(
            children: [
                Row(
                    children: [
                        Padding(
                            padding: const
                            EdgeInsets.symmetric(horizontal: 20),
                            child: CircleAvatar(
                                radius: 25,
                                backgroundImage:
                                    AssetImage(Assets.images.one),
                            ),
                        ),
                    ],
                    Column(
                        crossAxisAlignment:
                            CrossAxisAlignmentAlignment.start,
                        children: [
                            SizedBox(
                                width: 200,
                                child: Text(
                                    'Ботиралиев Бахтиер Баходиров',
                                    overflow: TextOverflow.ellipsis,
                                    style: AppTextStyles.body16wb,
                                ),
                            ),
                            const Text('точка продаж'),
                        ],
                    )
                ],
            ],
        )
    ],

```

```
        ),  
        const SizedBox(height: 20),  
        ListView.separated(  
            shrinkWrap: true,  
            itemBuilder: (context, index) => InkWell(  
                onTap: () {  
                    Navigator.pop(context);  
                    if (index == 5) {  
                        showDialog(  
                            context: context,  
                            builder: (context) =>  
                                UniverDialog(  
                                    gcontext: context,  
                                    title: 'Exit?',  
                                    yes: () {  
  
                        Navigator.pushReplacementNamed(  
                            context,  
                            Routes.login);  
                        },  
                    ));  
                } else {  
                    goToPage(index);  
                }  
            },  
            child: Container(  
                height: 40,  
                decoration: BoxDecoration(  
                    color:
```

```
        selDrawer == index ?  
        AppColors.black : null),  
            alignment: Alignment.center,  
            child: Text(  
                drawerMenu[index],  
                style:  
                    AppTextStyles.body17w5.copyWith(  
                        color: selDrawer == index  
                            ? AppColors.white  
                            : null),  
                    ),  
                    ),  
                    ),  
                    ),  
                    separatorBuilder: (context, index) => index  
                    == 2  
                    ? const Divider(  
                        height: 2,  
                        thickness: 2,  
                    )  
                    : const SizedBox(),  
                    itemCount: drawerMenu.length)  
                ],  
            ),  
        );
```