**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:** | **\_\_\_\_\_\_\_\_\_\_**  (подпись) | **Ботиралиев Б.Б.** |
|  |  | Студент группы **655 - 20 ПИ** |
| **Руководитель:** | **\_\_\_\_\_\_\_\_\_\_**  (подпись) | **Хусанов.Б** |

**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 FAST API для сервиса автоматизации управление торговли в магазинах и язык Dart и его фреймворк для кроссплатформенной-разработки такой как Flutter разработан мобильное приложение для этого FAST API. Выпускная квалификационная работа состоит из введения, теоретической части, проектной части, заключения, списка использованной литературы и приложения.

**ANNOTATION**

This graduation qualification work was carried out by a student of group 655-20 Botiraliev Bakhtiyor, in this thesis using for the server part the high-level programming language Python FAST API for the automation service for managing trade in stores and the Dart language and its framework for cross-platform development such as Flutter, a mobile application was developed for this FAST API. The final qualifying work consists of an introduction, a theoretical part, a project part, a conclusion, a list of references and an appendix.

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ 4**](#_gjdgxs)

[**I.**](#_30j0zll) **Теоретическая часть 6**

[1.1](#_1fob9te) Понятие автоматизации и ее значение 6

[1.2](#_3znysh7) Обзор существующих сервисов автоматизации публикаций 9

[1.3](#_2et92p0) Что такое Flutter 14

[1.4](#_tyjcwt) Обзор языка программирования Dart 22

[1.5](#_3dy6vkm) Основы [Dart](#_tyjcwt) [для разработки мобильных приложении 27](#_3dy6vkm)

[1.6](#_1t3h5sf) Основы работы с фреймворком Flutter 43

[**II.**](#_4d34og8) **Проектная часть 55**

[2.1](#_2s8eyo1) Создание технического задания проекта 55

[2.2](#_17dp8vu) Выбор платформы для создания пользовательского интерфейса 58

[2.3](#_3rdcrjn) Настройка среды разработки 59

[2.4](#_26in1rg) Создания пользовательского интерфейса 64

[2.5](#_lnxbz9) Сборка проекта и тестирование функциональности 70

[**Заключение 76**](#_35nkun2)

[**Список использованной литературы 77**](#_1ksv4uv)

[**Приложение 78**](#_44sinio)

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.

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

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

1. 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.

2. 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.

3. 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.

**Организация работы.** В дальнейшем в работе будет представлен обзор существующих сервисов автоматизации и основных принципов работы с Flutter. Затем будет проведен анализ требований к разрабатываемому сервису и представлено его архитектурное решение. После этого будет описан процесс разработки и реализации онлайн-сервиса автоматизации с использованием технологий Flutter. В заключении будут подведены итоги работы и предложены рекомендации по дальнейшему развитию сервиса.

Таким образом, данная дипломная работа имеет актуальность и важность в контексте современных требований к эффективной работе с торговлем, и ее результаты будут способствовать упрощению и оптимизации процессов торговли для пользователей платформы.

глава i. Теоретическая часть

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.

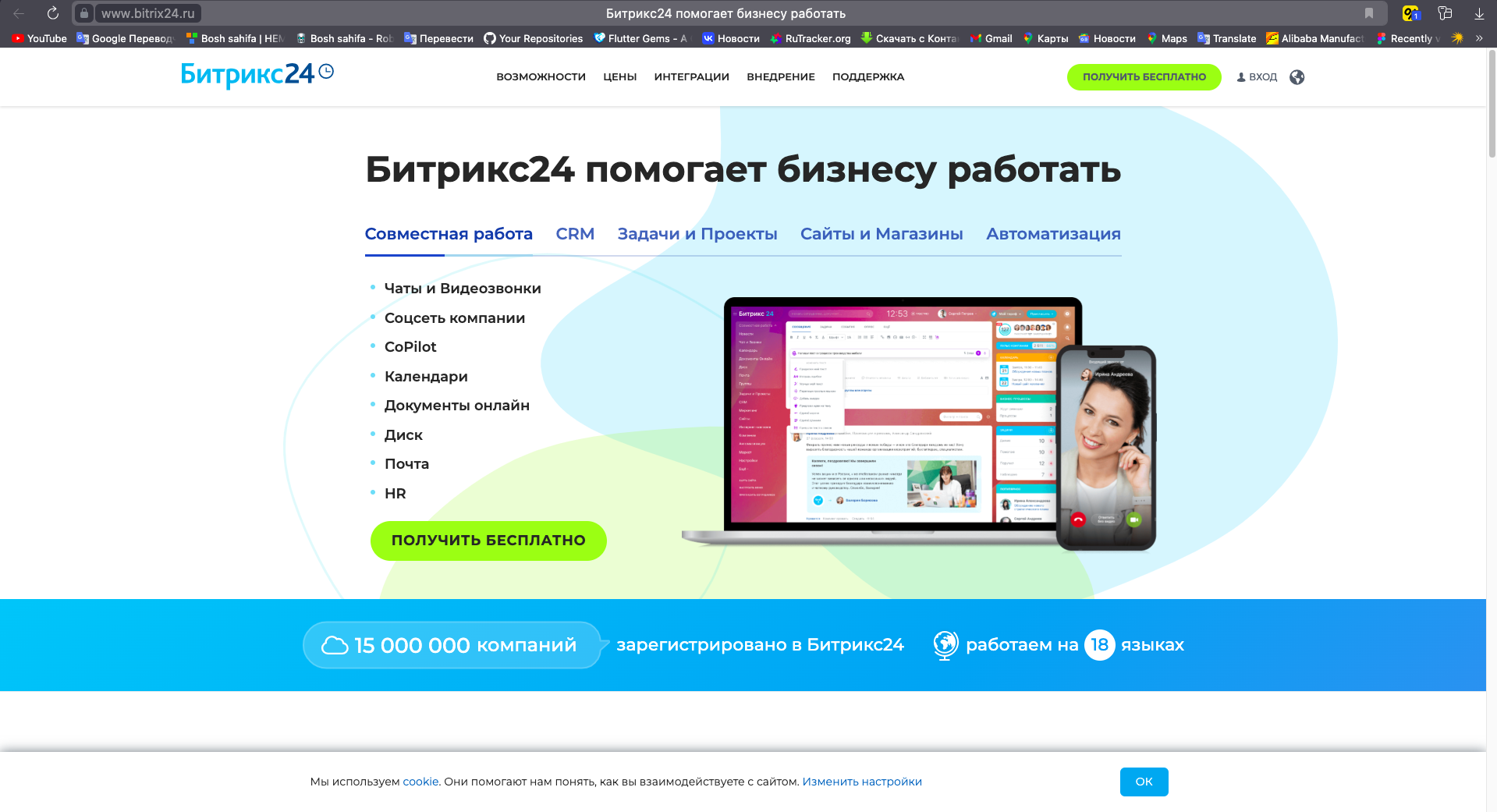


Рисунок 1.2.1

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.

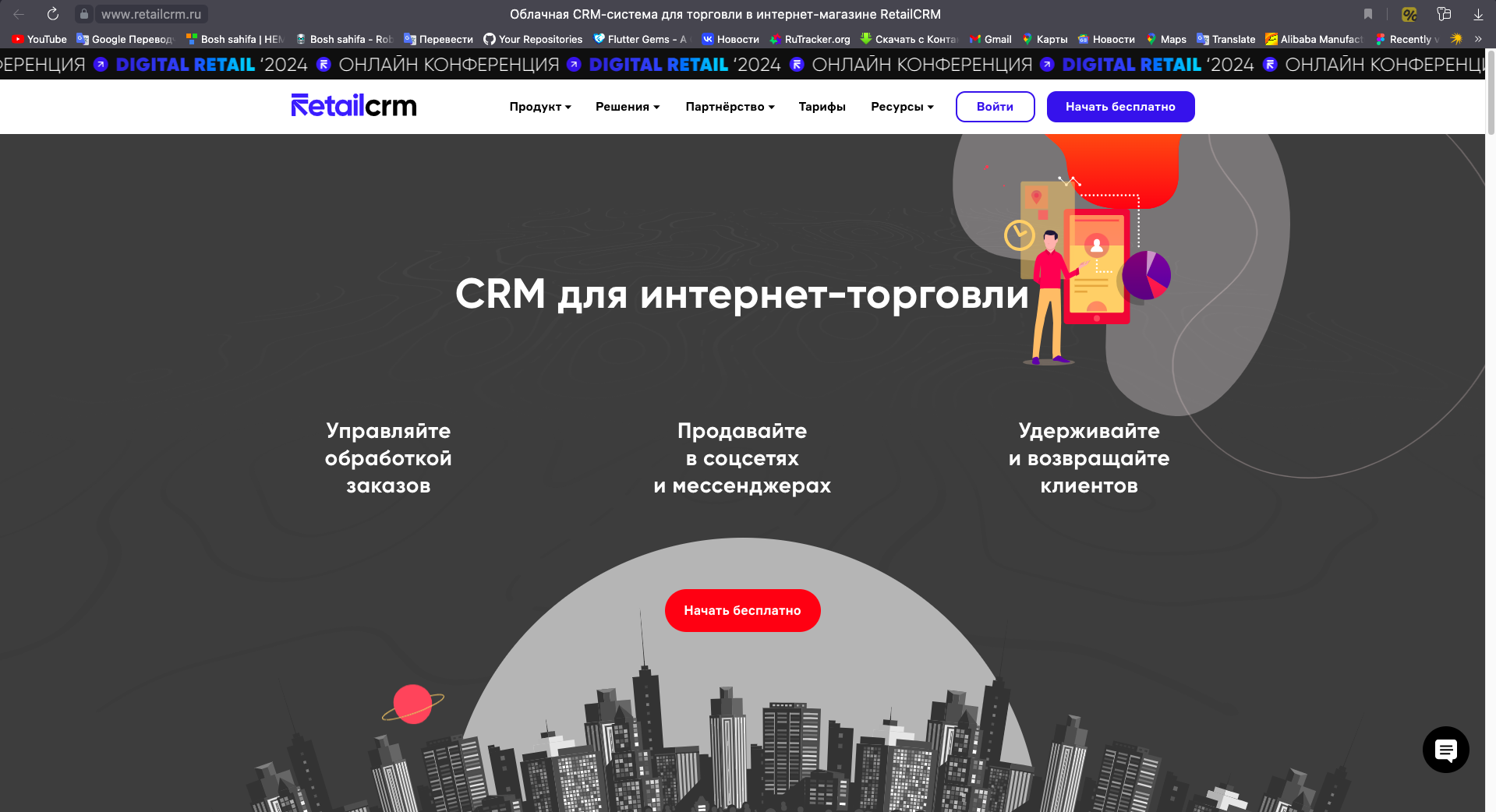


Рисунок 1.2.2

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 распространенность, 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.

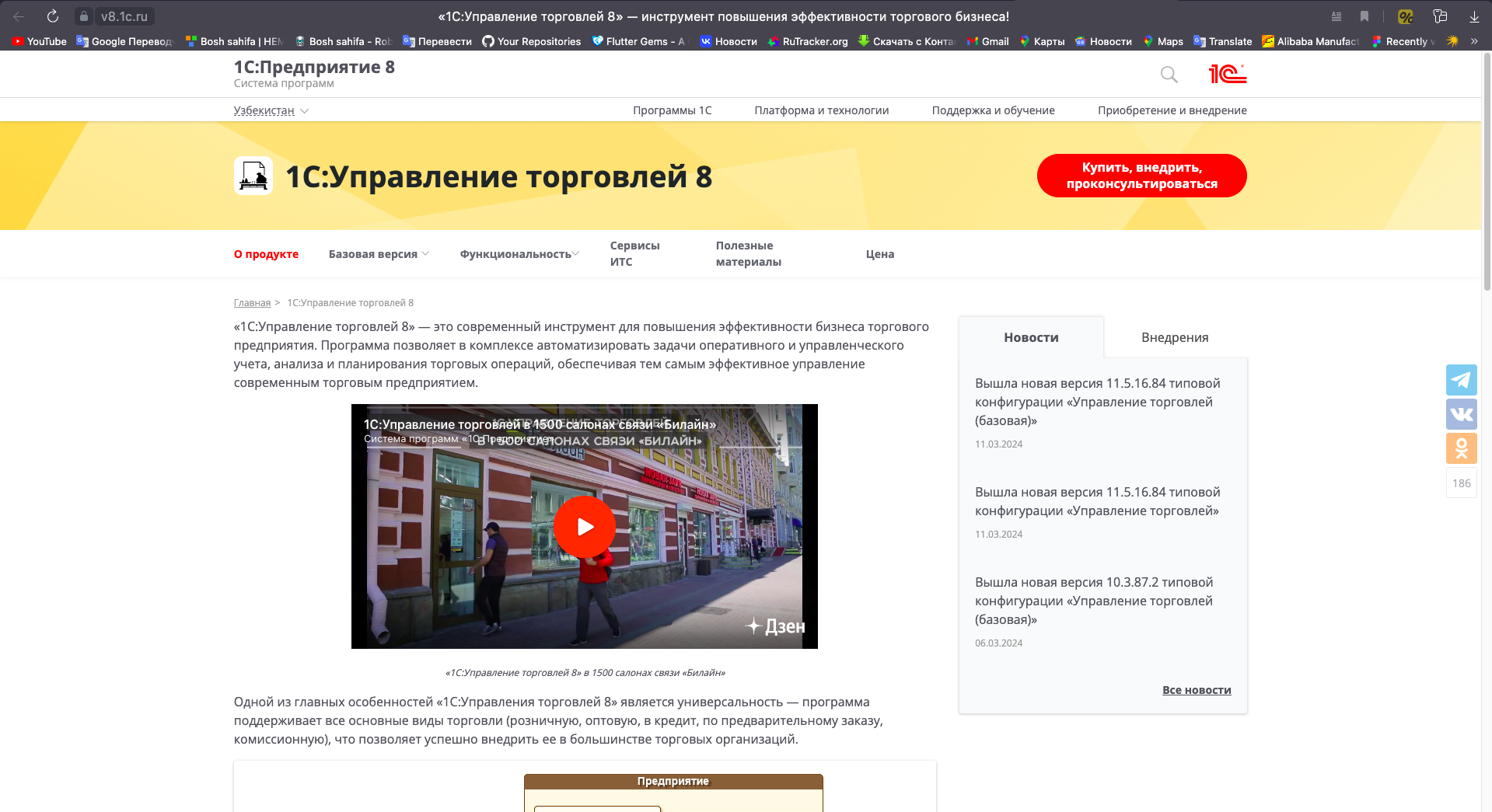


Рисунок 1.2.3

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.

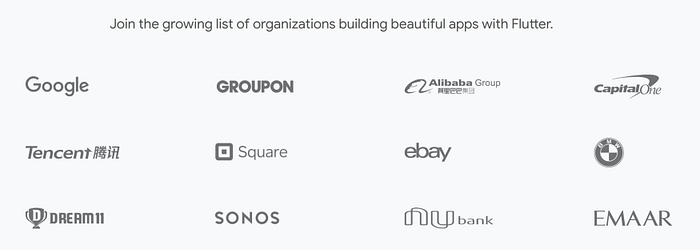


Рисунок 1.3.1

**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).

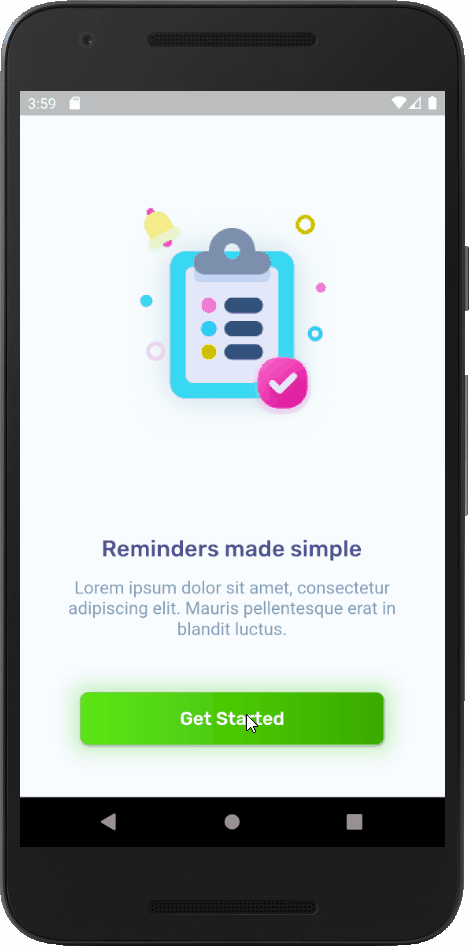


Рисунок 1.3.2

**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.

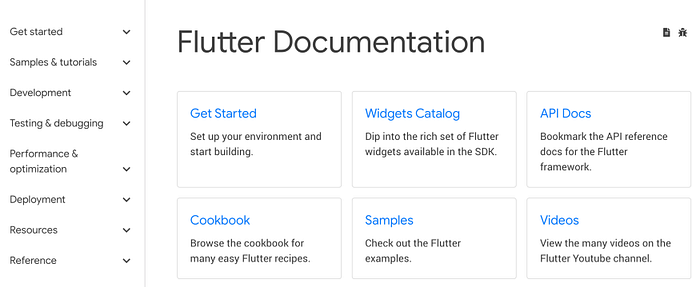


Рисунок 1.3.3

**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 С 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. AOT 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 Дарт 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 С++ 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 адептов 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 приложении.

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, булевы 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.

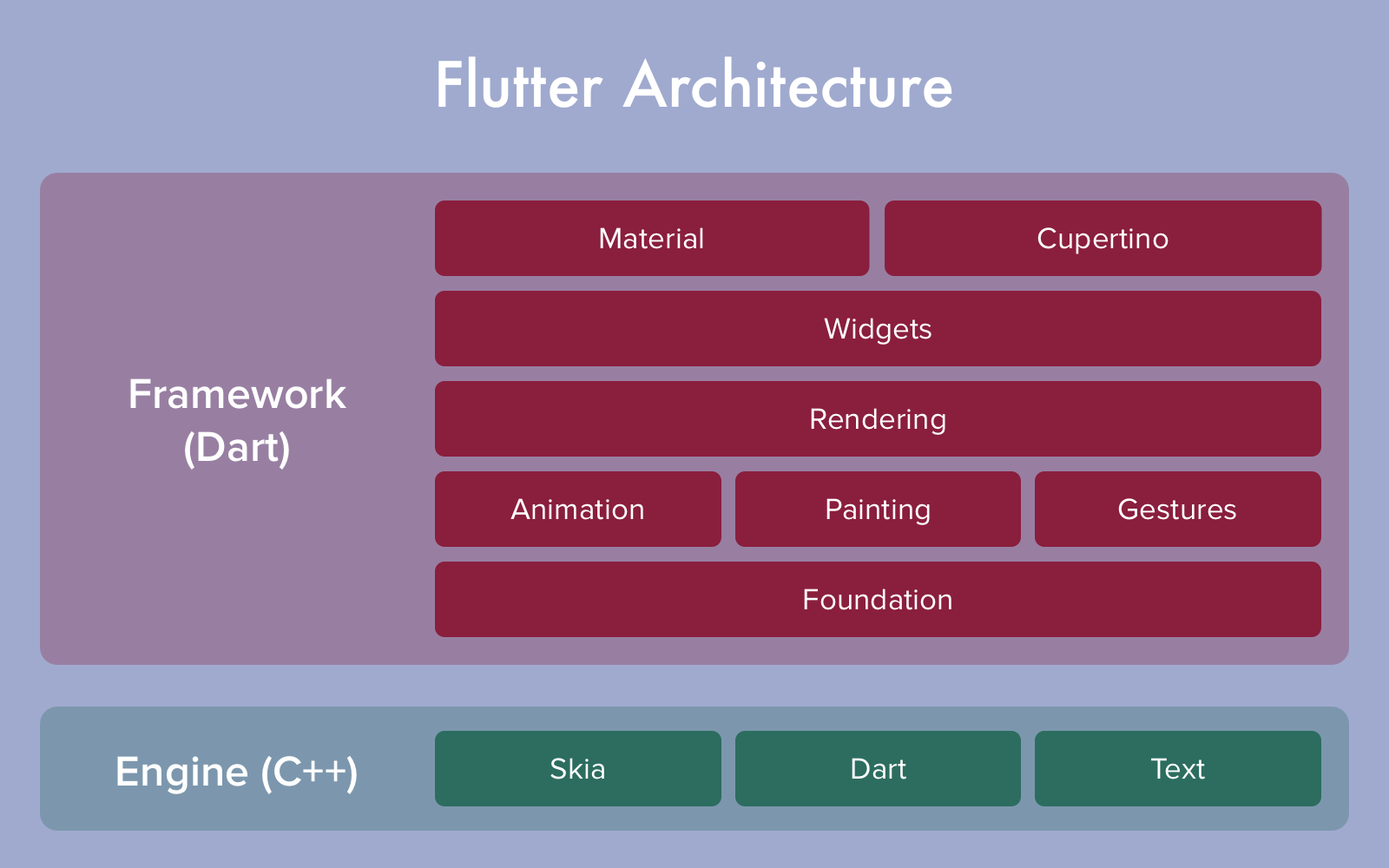
Приложения запускаются на основе комбинации движка рендеринга (построенного на C++) и Flutter (построенного на Dart). Все файлы, сгенерированные таким образом, присоединяются к каждому приложению и программному обеспечению сборки SDK для конкретной платформы.

Рисунок 1.6.1

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

x Flutter requires a minimum XCode version of 7.0.0.

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

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

Install via ‘brew install ideviceinstaller’.

x 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

Как видно, текст появился за статус-баром. Так произошло потому, что туда установлены координаты Flutter (0,0).

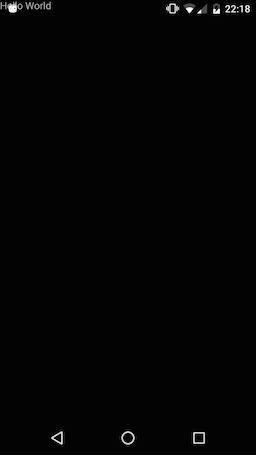


Рисунок 1.6.3

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(

new Padding(

padding: const EdgeInsets.only(top: 24.0),

child: new Text("Hello, World")

)

);

}

В примере выше создан виджет Padding, в котором отступы установлены на 24 при помощи объекта EdgeInsets, а в качестве дочернего элемента указан виджет Text. Запустите приложение и увидите, что текст теперь ниже.

Примечание: если вы знакомы с Java, то имейте в виду, const EdgeInsets.only(top: 24.0) всего лишь вызов конструктора EdgeInsets. Он возвращает экземпляр объекта, который будет константой времени компиляции. В этом разница между Java и Dart.

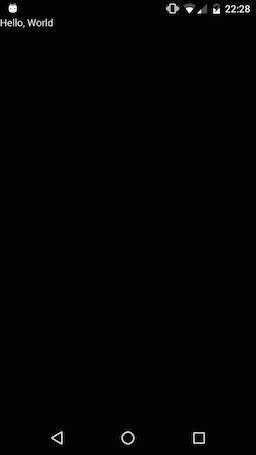


Рисунок 1.6.4

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

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

Глава II. Проектная часть

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 кросс-платформенных 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 платформах, 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.

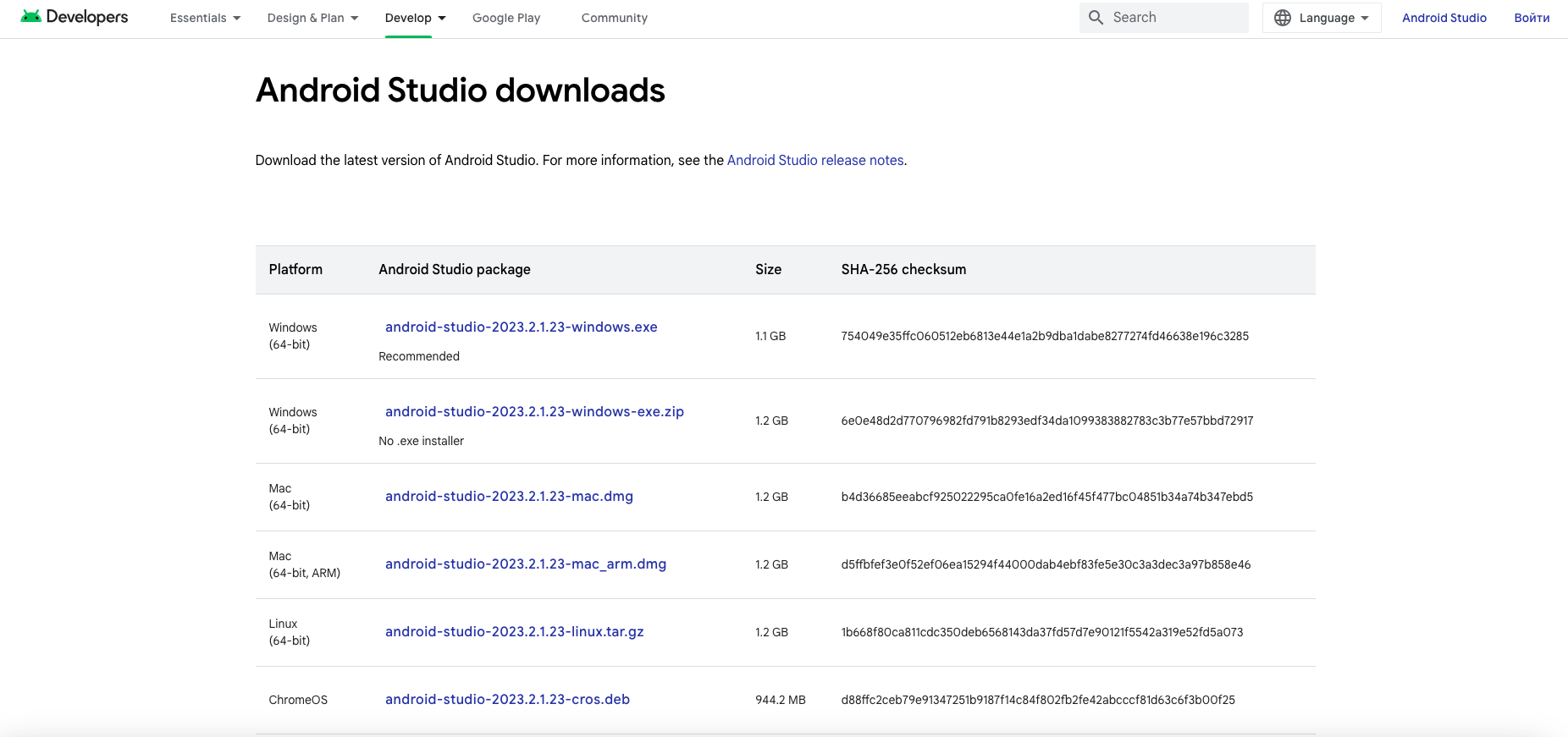


Рисунок 2.3.2

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.

Скачайте Flutter SDK

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

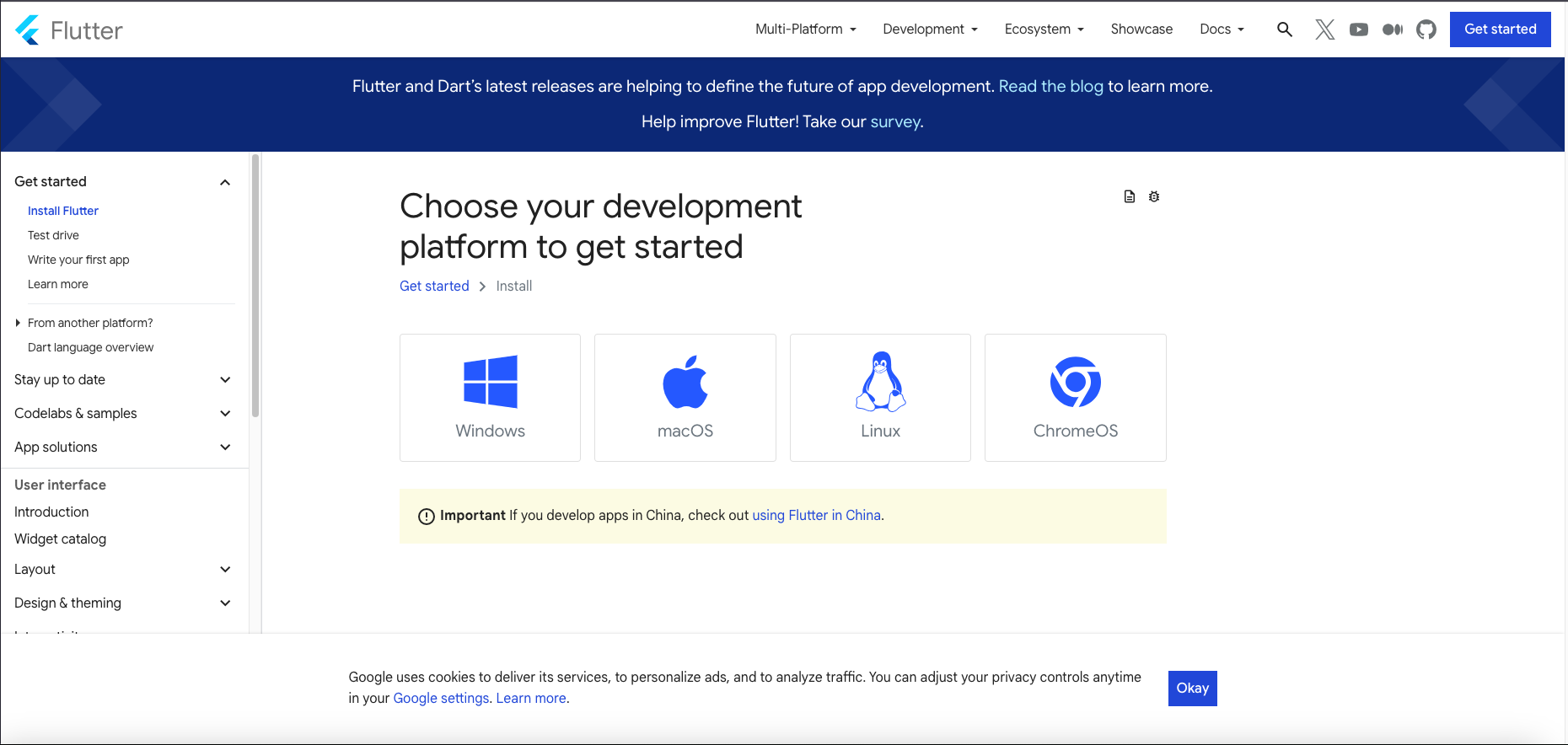


Рисунок 2.3.3

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.

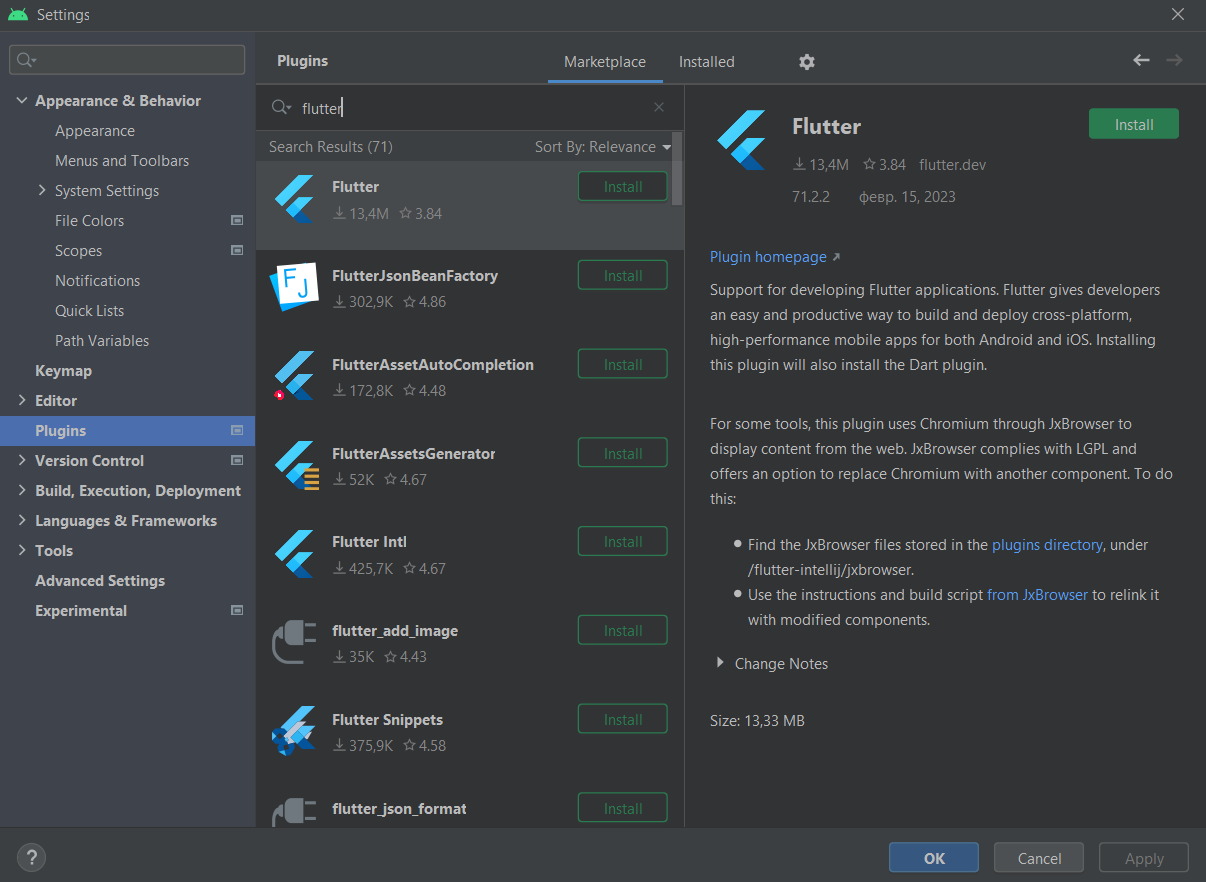


Рисунок 2.3.4

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.

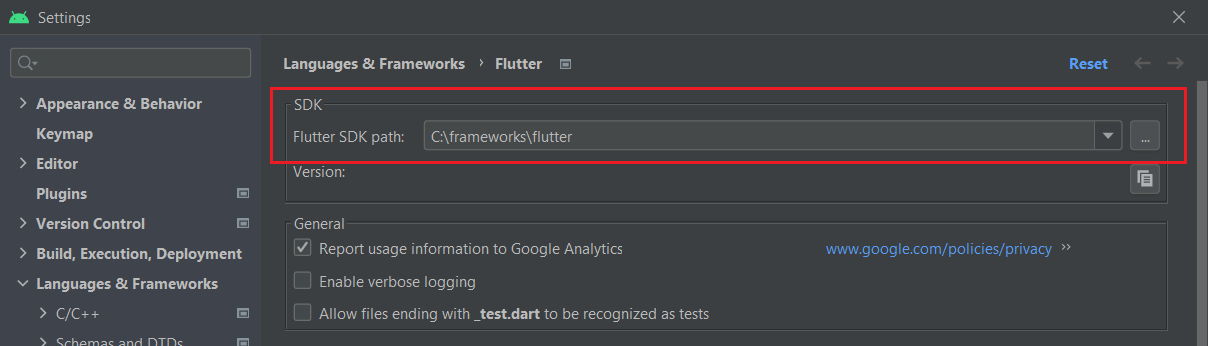


Рисунок 2.3.5

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.

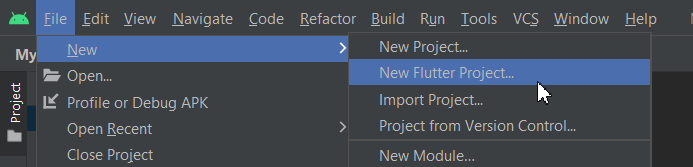


Рисунок 2.3.6

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:

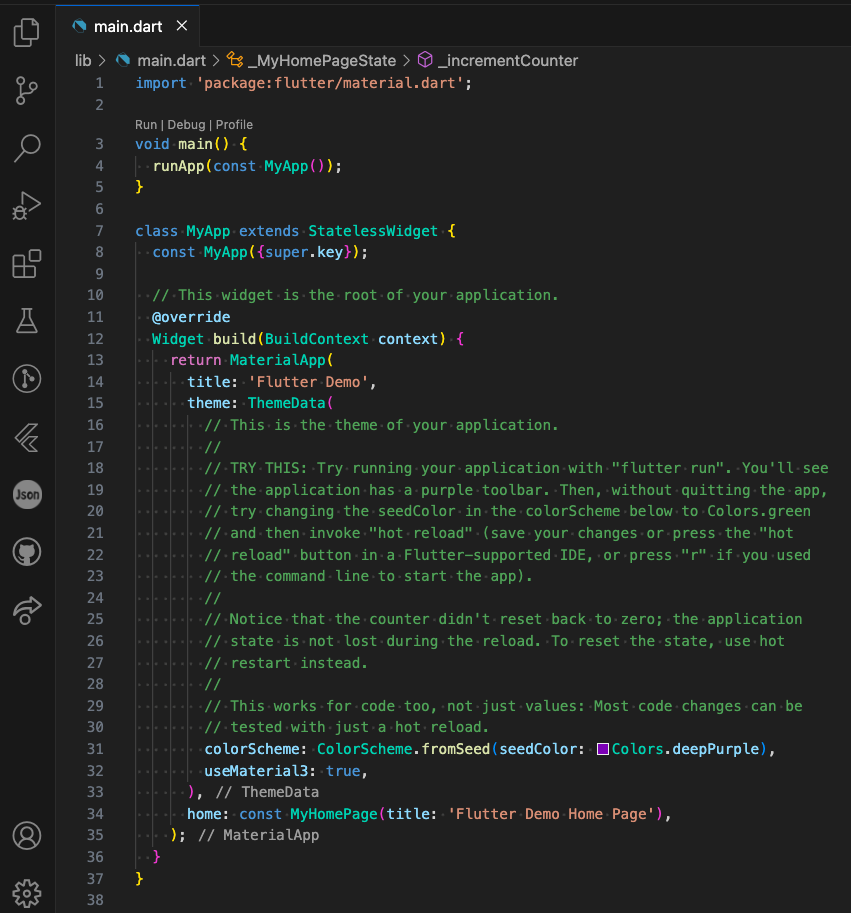


Рисунок 2.4.1

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

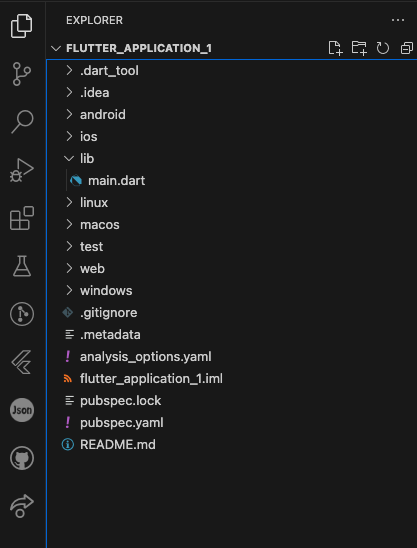


Рисунок 2.4.2 – Структура нового приложение

В директории lib создаём базовый файл и остальные файлы для нашего проекта. Эти шаблоны служат помогают нам писать код который не будет повторятся 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 помощь. Делаем разметку этого меню в файле routes.dart чтобы она и в других окнах отображалась.

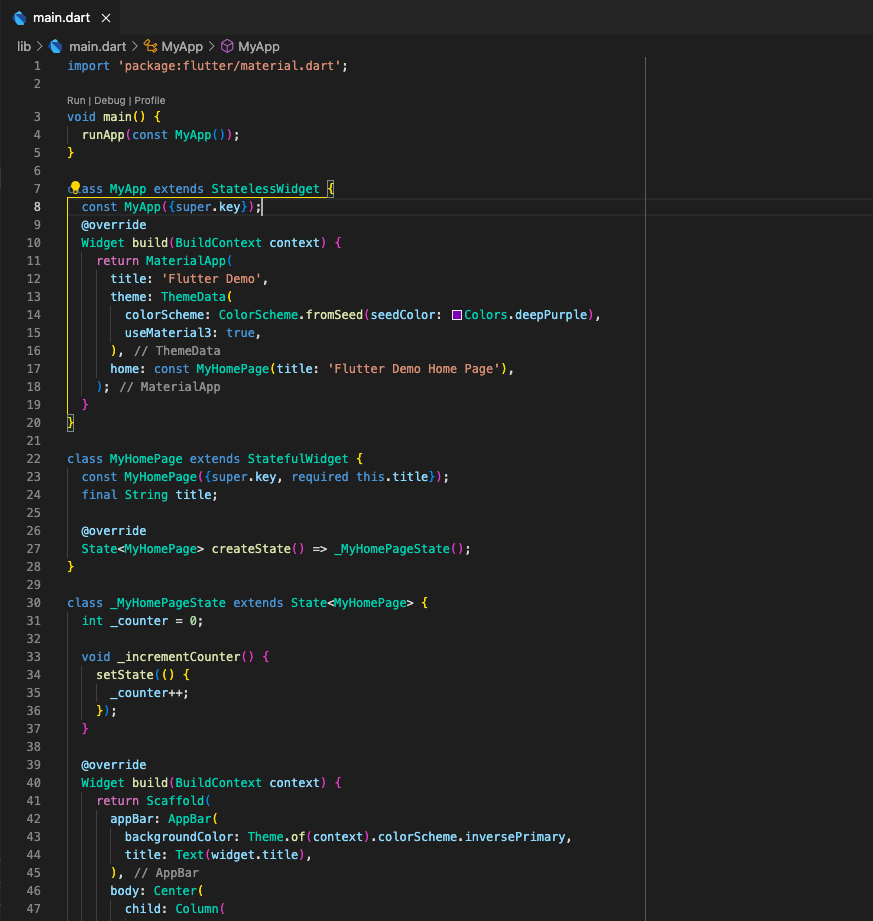


Рисунок 2.9 – Основная страница приложение

## 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.

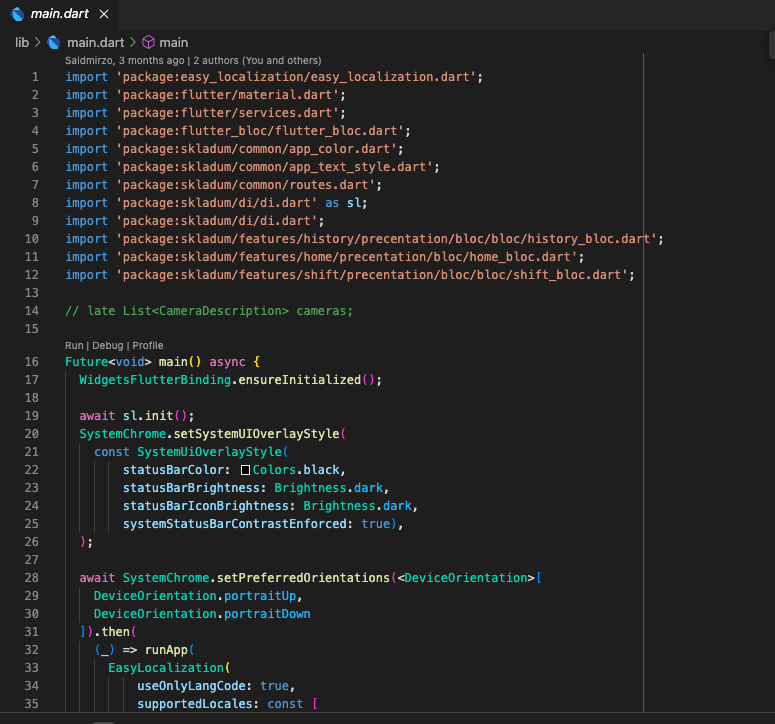


Рисунок 2.13 – Пример основного файла.

Теперь нам нужно создать ещё один файл с названием home.dart она нужна нам чтобы объеденить всех экранов и показать как выглядеть на мобильном устройстве . Здесь мы будем запускать приложение с нашим проекта и базы данных для нашего проекта. И в routes.dart файле мы будем указывать порядок их запуска и требования контейнеров друг-другу.

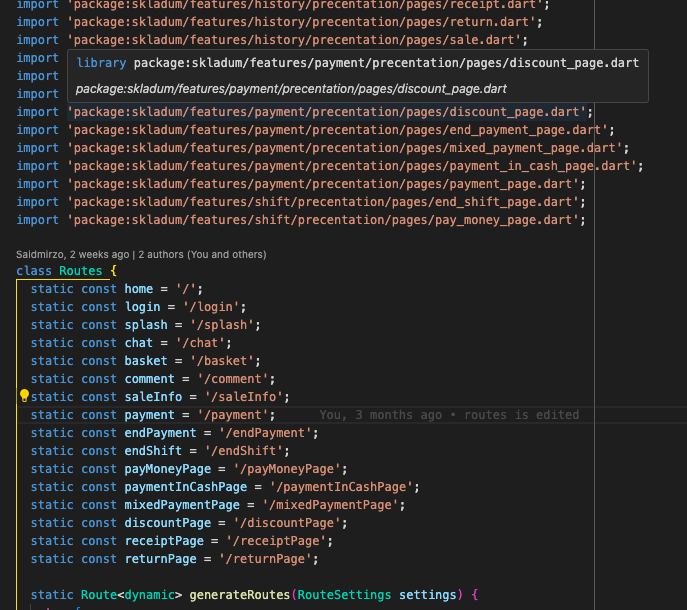


Рисунок 2.14 – Пример файла навигации.

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.

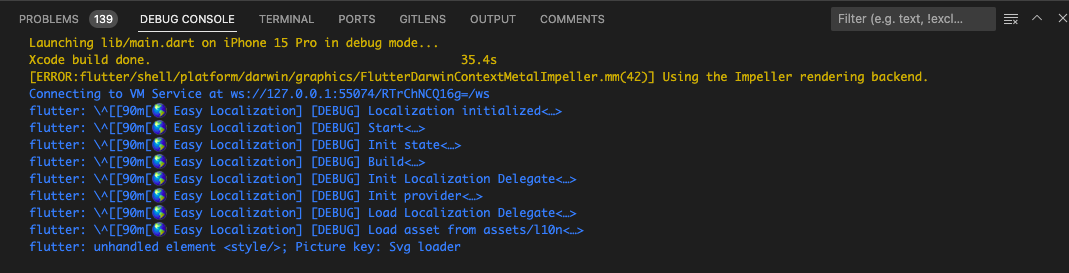


Рисунок 2.15 – Запуск проекта.

**3. ОФОРМЛЕНИЕ РЕЗУЛЬТАТОВы 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.



Рис.3.1.1. Splash screen

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

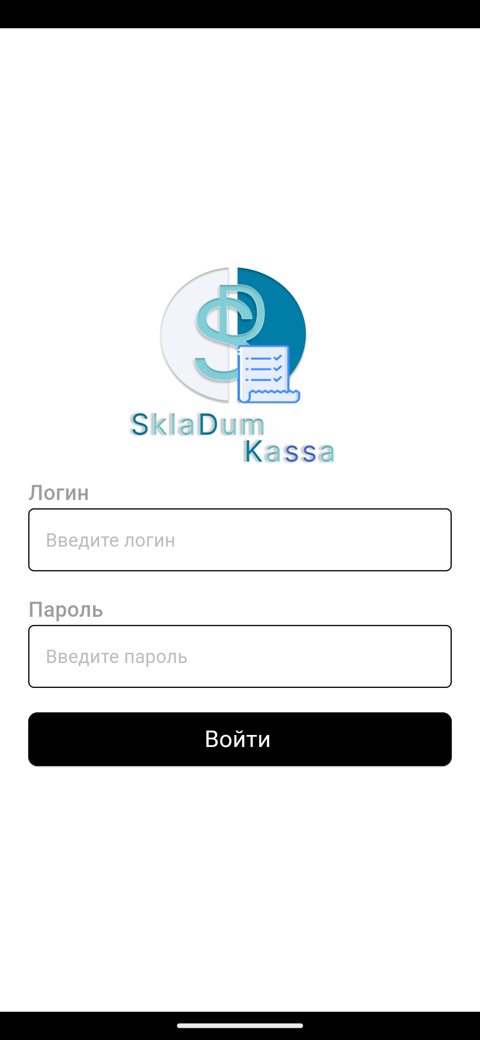


Рис.3.1.3. Логин

**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.

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



Рис.3.1.4. Главная страница

**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 устройство(внешний сканнер) и экономить деньги.

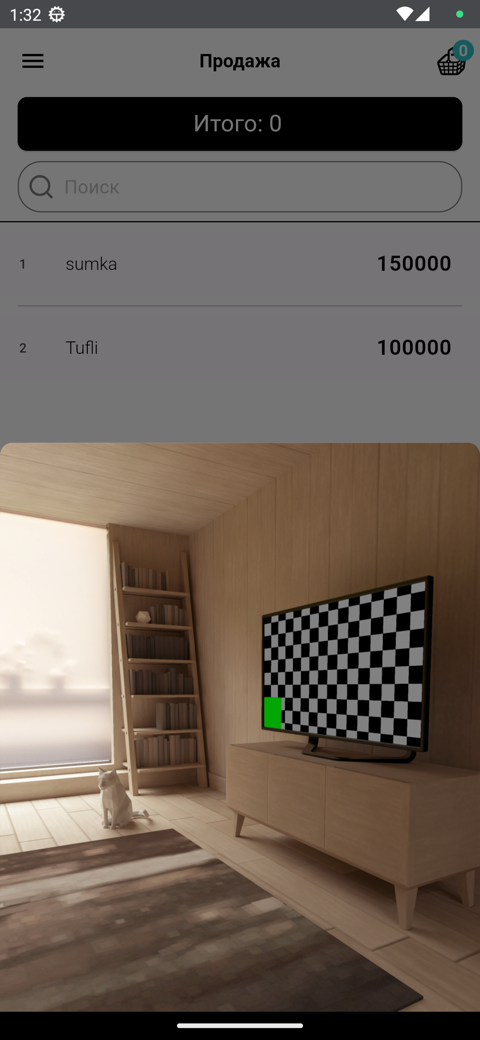


Рис.3.1.5. Сканер

**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 заполняет поля этим кодом нам остается ввести имя и цену на товар.

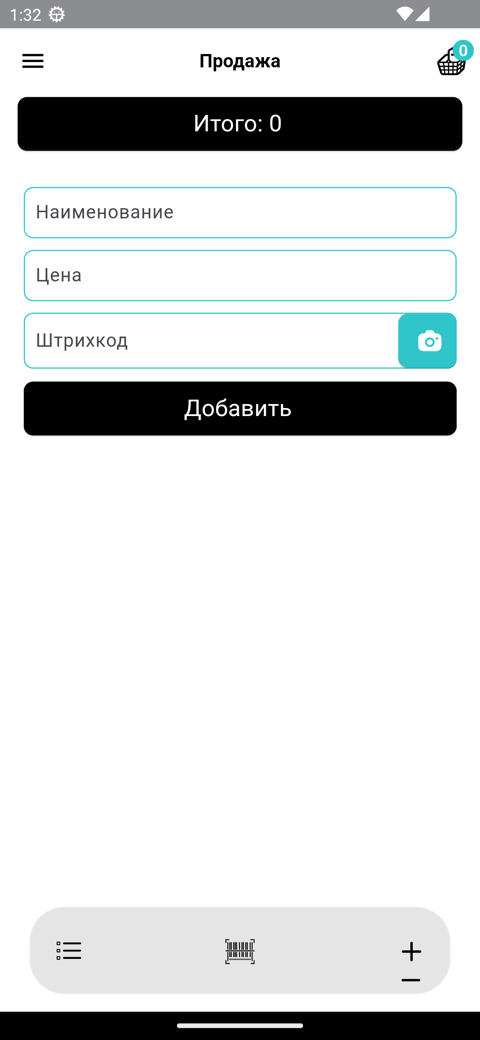


Рис.3.1.7.Добавить товар

**4. ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ**

**4.1. Расчет искусственного освещения методом коэффициента светового потока.**

**Назначение.** Расчет общей освещенности пола шириной А м и (высотой) В м люминесцентными лампами: высота белого потолка Нм Rп = 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.**

=20\*1,5\*432\*0,95/8,67\*30=47,3

Здесь Fl – световой поток каждой лампы, лм;

Эм – норма освещенности, лк;

К – коэффициент резерва (табл. 4.1.1)

S – площадь помещения, м2

N – количество ламп

- коэффициент использования светового потока, то есть общий световой поток во всех помещениях в зависимости от токов, падающих на рабочую поверхность. (Таблица 4.1.2).

Z- отношение средней освещенности к минимальной освещенности

Z = 1,15 ÷ 1,2

1. Определение единицы измерения помещения.

=432/6\*(18+24)=1,71

Здесь: S – площадь помещения, 432м2.

А - 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 =l,5; Л = 1,5h=1,5 \* 4,7=7,05

4. Определение количества светильников по ширине и длине помещения:

=18/7,05=2,55=24/7,05=3,4

5. Общее количество светильников определяется следующим образом:

*Н=*Н(А)*Н (Б) =2,55\*3,4=8,67*

6. Определить световой поток каждой лампы.

Если каждая лампа Фл превышает расчетную отметку светового потока (Ф3), количество ламп следует пересчитать.

**Начальные данные.** K - резервный коэффициент

**Таблица 4.1.1**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 4.1.2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | | |
| Rp разными символами в рупиях (%). | | | | | | | | | | | |
|  | 30  10 | 50  30 | 70  50 | 50  30 | 50  50 | 70  50 | 70  70 | 50  30 | 50  50 | 50  50 | 70  70 |
| 0,5  0,6  0,7  0,8  0,9  1.0  1.1  1,25  1,5  1,75  2.0  2.25  2,5  3.0  3,5  4.0  5.0 | 15  19  23  25  27  29  30  31  33  34  36  37  38  39  40  41  42 | 17  22  25  27  29  30  31  33  34  36  37  38  39  40  41  42  43 | 20  24  27  30  31  32  33  35  37  38  39  41  42  43  44  45  46 | 13  16  19  21  22  23  24  26  28  29  31  32  33  35  36  36  39 | 15  19  21  23  24  25  27  28  30  32  33  35  36  37  38  39  41 | 17  22  24  26  28  30  28  30  31  33  35  37  39  40  46  47  49 | 23  17  30  32  33  35  36  38  40  41  43  44  46  47  49  50  52 | 13  16  19  21  22  23  24  25  26  27  28  29  30  31  32  32  33 | 15  19  21  22  23  24  25  26  27  28  29  30  31  32  32  33  34 | 15  19  21  23  24  24  25  26  28  29  30  31  32  33  33  34  34 | 19  22  24  25  26  27  28  29  30  31  32  33  33  34  34  35  35 |

**4.2. Варианты решения задачи.**

**Таблица 4.1.3**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Параметры | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Эм, лк | | 500 | 300 | 400 | 200 | 300 | 500 | 200 | 400 | 300 | 200 | 500 | 400 | 300 | 500 | 200 |
| Размеры помещения | Высота, м Ширина, м  Высота, м | 15  10  5 | 25  15  6 | 70  50  7 | 50  30  6 | 70  35  7 | 25  15  5 | 24  18  6 | 30  18  7 | 24  12  6 | 18  12  9 | 10  8  4 | 12  10  5 | 14  11  3 | 20  25  4 | 30  25  6 |
| Параметры | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| Эм, лк | | 300 | 400 | 200 | 500 | 300 | 500 | 200 | 400 | 300 | 200 | 500 | 400 | 300 | 500 | 200 |
| Размеры помещения | Высота, м Ширина, м  Высота, м | 28  20  7 | 23  10  5 | 40  30  4 | 3528  7 | 45  40  8 | 25  15  5 | 24  18  6 | 30  18  7 | 24  12  6 | 18  12  9 | 10  8  4 | 12  10  5 | 14  11  3 | 20  25  4 | 30  25  6 |

**4.3. Вопросы для самостоятельной подготовки.**

1. Что такое световой поток?   
Световой поток — это мера световой энергии, испускаемой источником света. Измеряется в люменах (лм). Этот параметр отражает общее количество света, выходящего из источника. Важен для определения эффективности осветительных приборов. Помогает выбрать нужные лампы для различных условий освещения.

2. Что такое сила света?

Сила света описывает, насколько ярко светит лампа в определённом направлении. Измеряется в канделах (кд). Является важным показателем при проектировании осветительных систем. Сила света показывает распределение света в пространстве. Это ключевой фактор при создании равномерного освещения.

3. Каковы качественные показатели освещения?

Качественные показатели освещения включают равномерность распределения света, контрастность и цветопередачу. Также важно отсутствие мерцания, чтобы избежать утомляемости глаз. Хорошая цветопередача необходима для точного восприятия цветов. Равномерное освещение способствует созданию комфортной среды для работы и отдыха. Эти параметры определяют не только комфорт, но и безопасность пребывания в помещении.

4. Что такое естественное и искусственное освещение?

Естественное освещение обеспечивается солнечным светом через окна, люки и другие открытые поверхности. Искусственное освещение создается с помощью ламп и светильников. Оба типа освещения используются для удовлетворения потребностей человека в свете. Искусственное освещение позволяет контролировать интенсивность и направленность света. Использование обоих видов освещения должно быть сбалансированным для обеспечения эффективности и комфорта.

5. Какие светильники вы знаете?

Существуют различные типы светильников, включая люминесцентные, светодиодные, галогенные и лампы накаливания. Каждый тип имеет свои особенности и предназначен для определенных задач. Люминесцентные лампы эффективны и долговечны. Светодиодные лампы отличаются низким энергопотреблением и длительным сроком службы. Галогенные лампы обеспечивают яркий, направленный свет. Лампы накаливания широко используются благодаря своему теплому свету.

6. Какие конкретные виды работ вы знаете?

Среди видов работ, требующих особого освещения, можно выделить чтение, письмо, выполнение точных механических работ, чертежи и дизайн. Для каждого вида работы требуются различные уровни освещенности и качества света. Например, для чтения и письма нужен яркий, равномерный свет без мерцания. Для точных механических работ важно точное и направленное освещение.

7. Какие требования к освещению на производстве?

На производстве требования к освещению включают обеспечение достаточной интенсивности и качества света для выполнения работы без риска для здоровья. Необходимо учитывать равномерность распределения света и минимизацию отраженного блеска. Также важна адаптация освещения под разные виды деятельности. Высокое качество освещения способствует повышению продуктивности и снижению утомляемости работников.

8. Сколько должно быть аварийное освещение?

Аварийное освещение должно обеспечивать минимально необходимый уровень видимости для безопасной эвакуации людей в случае аварии или отключения основного освещения. Обычно это 10-30% от нормы основного освещения, достаточно для того, чтобы безопасно покинуть помещение.

9. Какие лампы используются для освещения взрывоопасных помещений?

В взрывоопасных помещениях используются специальные взрывобезопасные лампы, которые соответствуют строгим стандартам безопасности. Эти лампы разработаны таким образом, чтобы предотвратить возможность взрыва даже при повреждении светильника.

# **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**

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

Список интернет ресурсов:

* 1. [flutter.dev (flutter.io)](https://flutter.su/note/57#flutter.dev) [En]
  2. [dart.dev](https://flutter.su/note/57#dart.dev) [En]
  3. [Flutter for web](https://flutter.su/note/57#Flutter-for-web) [En]
  4. [Flutter for desktop](https://flutter.su/note/57#Flutter-for-desktop) [En]
  5. [Flutter Codelabs](https://flutter.su/note/57#Flutter-Codelabs) [En]
  6. [Flutter Community (Medium)](https://flutter.su/note/57#Flutter-Tutorials-Handbook) [En]
  7. [Flutter Tutorials Handbook](https://flutter.su/note/57#Flutter-Tutorials-Handbook) [En]
  8. [Flutter by Google](https://flutter.su/note/57#Flutter-by-Google) [En]
  9. [Уроки Flutter](https://flutter.su/note/57#%D0%A3%D1%80%D0%BE%D0%BA%D0%B8-Flutter) [Ru]

# **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/precentation/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/precentation/pages/history.dart';**

**import 'package:skladum/features/home/data/model/product\_model.dart';**

**import 'package:skladum/features/home/precentation/bloc/home\_bloc.dart';**

**import 'package:skladum/features/home/precentation/pages/add.dart';**

**import 'package:skladum/features/home/precentation/pages/products.dart';**

**import 'package:skladum/features/home/precentation/pages/search.dart';**

**import 'package:skladum/features/home/precentation/widgets/scaner\_widget.dart';**

**import 'package:skladum/features/login/presentation/widgets/tabbar.dart';**

**import 'package:skladum/features/settings.dart';**

**import 'package:skladum/features/shift/precentation/pages/shift\_page.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: CrossAxisAlignment.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(**

**drewerMenu[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: drewerMenu.length)**

**],**

**),**

**);**