

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

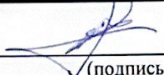
Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

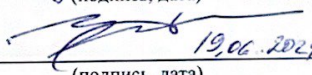
на тему

**РАЗРАБОТКА МОБИЛЬНОГО
ПРИЛОЖЕНИЯ ДЛЯ КАФЕ**

Студент

 19.06.2024 Киреев Б.П.
(подпись, дата)

Руководитель

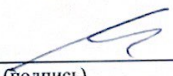
 19.06.2024 Зинкин С.А.
(подпись, дата)

Нормоконтролёр

 19.06.2024 Бычков А.С.
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой

 Митрохин М.А.
(подпись)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

20.06.2024 № 30)

Секретарь ГЭК

 Gurin E.I.
(подпись)

Пенза 2024

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«19» апреля 2024 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Киреев Богдан Павлович гр. 20ВВП1 факультета Вычислительной техники.

2. Тема работы: «Разработка мобильного приложения для кафе»

Тема утверждена приказом ПГУ № 405/о от "19" апреля 2024 г.

3. Руководитель бакалаврской работы профессор каф. ВТ ПГУ Зинкин С.А.

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

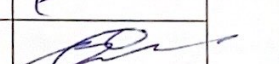
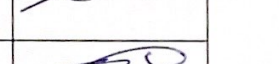
Задание на работу (назначение разработки, исходные данные и т.п.)

Необходимо спроектировать и разработать кроссплатформенное мобильное приложение кафе для заказа блюд. Мобильное приложение должно предоставлять возможность со стороны пользователя выбирать блюда, детально их изучать и добавлять их в корзину для дальнейшего заказа.

5. Перечень подлежащих разработке вопросов

Необходимо провести анализ предметных областей: изучить особенности разработки кроссплатформенных мобильных приложений. Важным аспектом является изучение возможных альтернатив, а также методов разработки мобильных приложений с использованием фреймворков, таких как Flutter, и облачной базы данных Firebase.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	20	26.04.2024 – 27.04.2024	
2	Анализ требований	32	28.04.2024 – 30.04.2024	
3	Выбор инструментов, применяемых при проектировании и разработке	30	01.05.2024 – 03.05.2024	
4	Разработка архитектуры проекта	16	04.05.2024 – 5.05.2024	
5	Разработка клиентской части мобильного-приложения	122	6.05.2024 – 19.05.2024	
7	Тестирование	50	19.05.2024 – 23.05.2024	
8	Оформление пояснительной записки	54	24.05.2024 – 31.05.2024	

Дата выдачи задания "19" апреля 2024 г.

Руководитель бакалаврской работы


(подпись, дата)

Зинкин С.А.

Задание к исполнению принял студент


(подпись, дата)

Киреев Б.П.

Работу к защите допустить

Декан факультета


(подпись, дата)

19.06.2024

Фионова Л.Р.

Реферат

Бакалаврская работа содержит 104 листа, 13 рисунков, 5 таблиц, 5 источников, 3 приложения

IOS, ANDROID, ОБЛАЧНАЯ БАЗА ДАННЫХ, FLUTTER, FIREBASE

Цель работы – разработать кроссплатформенное мобильное приложение для кафе, позволяющее пользователям оформлять выбирать блюда и в дальнейшем их заказывать.

В ходе разработки необходимо реализовать следующие возможности для пользователя: авторизация и регистрация, просмотр меню, *выбор блюда, подробное изучение блюда и добавление его в корзину.

В процессе разработки проводился анализ актуальности проекта, аналогов, основных платформ разработки ПО.

В результате проведенной работы создано мобильное приложение, соответствующее поставленной задаче.

					ПГУ1.090301.08.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Киреев Б.П.		19.06.21	Лит.	Лист	Листов
Провер.		Зинкин С.А.		19.06.21		2	104
Н. Контр		Бычков А.С.		19.06.21	ФВТ, каф. ВТ		
Утверд.		Митрохин М.А.		19.06.21	гр. 20ВВП1		
					Разработка мобильного приложения для кафе		
					Пояснительная записка		

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 Мобильное приложение для кафе	8
1.1 Анализ предметной области	8
1.1.1 Обзор аналогов	9
1.1.2 Приложение Дайкон	9
1.1.3 Приложение Кофейный экспресс	10
1.1.4 Приложение Вертел.....	11
1.1.5 Постановка задачи	11
1.2 Бизнес модель кафе	12
1.2.1 Общая концепция	12
1.2.2 Меню	13
1.2.3 Цифровые технологии	14
1.3 Выбор технологий разработки.....	14
1.3.1 Язык программирования Swift.....	14
1.3.2 Язык программирования Kotlin	15
1.3.3 Фреймворк Flutter	15
1.4 Выбор среды разработки.....	16
1.4.1 Visual Studio Code	16
1.4.2 Xcode	17
1.4.3 Android Studio	18
1.4.4 Firebase	19
1.5 Сравнение ОС	20
1.5.1 ОС macOS.....	20
1.5.2 ОС Windows	21
1.5.3 ОС Linux.....	22
1.6 Проектирование мобильного приложения.....	23
1.6.1 Архитектура приложения.....	23

1.6.2 Проектирование базы данных.....	27
1.7 Реализация приложения	28
1.7.1 Реализация регистрации и аутентификации пользователя.....	28
1.7.2 Реализация меню	39
1.7.3 Реализация карточки блюда	46
1.7.4 Реализация корзины	51
2 Безопасность жизнедеятельности.....	56
2.1 Потенциально опасные факторы при работе с ПЭВМ	56
2.2 Классификация помещения	57
2.3 Санитарно-гигиеническая характеристика	57
2.4 Расчет освещенности рабочего места.....	58
Заключение	61
Список использованных источников	62
Приложение А – UML-диаграммы.....	63
Приложение Б – Листинг программы	68
Приложение В – Презентация	96

Перечень сокращений и обозначений

В настоящей работе применяют следующие обозначения и сокращения

БД – база данных

ОС – операционная система

ПК – персональный компьютер

BLoC (Business Logic Component) – компонент бизнес-логики

IDE (Integrated Development Enviroment) – интегрированная среда разработки

UI (User Interface) – пользовательский интерфейс

UML (Unified Modeling Language) – унифицированный язык моделирования

UX (User Experience) – пользовательский опыт

					<i>ПГУ1.090301.08.001 ПЗ</i>	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Сегодня рынок мобильных приложений быстро расширяется; всё больше людей активно пользуются интернетом и проводят значительное количество времени в приложениях. Современные продвинутые модели мобильных устройств по своим функциям и характеристикам почти не уступают полноценным персональным компьютерам и ноутбукам, процессоры Apple Silicon на мобильной архитектуре ARM, которые используются в современных MacBook, отличное тому подтверждение. Сейчас почти не осталось людей, которые бы не пользовались мобильными устройствами, ведь их популярность объясняется функциональностью, доступом ко всем видам связи и, конечно, компактным размером.

Мобильное приложение является одним из ключевых инструментов для увеличения прибыли бизнеса. Такие приложения включают в себя различные компоненты и экраны, предназначенные для выполнения поставленных задач. Благодаря автоматизации и удобному интерфейсу, процесс взаимодействия с бизнесом через мобильное приложение получается простым и легким.

Сфера общественного питания не стала исключением, все больше заведений стремятся внедрять мобильные технологии для улучшения обслуживания клиентов и оптимизации своей деятельности. Например, многие рестораны предлагают своим посетителям возможность делать заказы через мобильное приложение, что значительно сокращает время ожидания и минимизирует вероятность ошибок при приеме заказа. Кроме того, мобильные приложения позволяют автоматизировать процессы бронирования столиков, управления очередями, а также запускать программы лояльности и предлагать клиентам индивидуализированные предложения.

Цели выпускной квалификационной работы:

- освоение актуальных и современных средств программной разработки;
- изучение нюансов разработки мобильных приложений для заведений быстрого питания;

					ПГУ1.090301.08.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

– реализация мобильного приложения необходимого для осуществления заказа в сети быстрого питания.

Для достижения поставленных целей, необходимо обозначить следующие задачи, решаемые в ходе выполнения выпускной квалификационной работы:

- провести анализ существующих мобильных приложений;
- провести анализ архитектуры мобильного приложения;
- спроектировать карту действий мобильного приложения;
- реализовать мобильное приложение;
- протестировать мобильное приложение.

					ПГУ1.090301.08.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1 Мобильное приложение для кафе

1.1 Анализ предметной области

Начало разработки продукта всегда связано с анализом предметной области.

Необходимо определить кто будет пользоваться продуктом, выбрать технологии, которые позволят в будущем поддерживать и развивать продукт.

В данном разделе рассматриваются общие теоретические вопросы, связанные с проектированием и реализацией мобильного приложения для кафе. Изучение данных вопросов поможет определить основные проблемы и задачи для разработки продукта.

Предметная область разработки мобильного приложения относится к области информационных технологий и включает в себя разработку программного обеспечения для взаимодействия бизнеса с потребителем.

На сегодняшний день мобильные приложения являются эффективным инструментом для продажи товаров и услуг. В нашем случае приложение способно продавать позиции из меню заведения. Это удобно, ведь покупателю не нужно будет стоять в очереди, что в результате сократит время на получение заказа и, следовательно, повысит лояльность клиента. Также появляется возможность более внимательно изучить меню, ознакомиться с калоражем блюд и даже совершить заказ вне заведения или даже в доставку.

Ниже предоставлены следующие аспекты, которыми должно обладать современное мобильное приложение для кафе:

– функциональность и UX. Одним из главных аспектов является обеспечение разнообразных функций, позволяющих покупателям быстро и безопасно регистрироваться, удобно просматривать меню блюд, подробно изучать отдельные позиции и осуществлять их добавление в корзину для дальнейшего оформления;

– безопасность и конфиденциальность. Важно обеспечить высокий уровень защиты данных и конфиденциальности аккаунта, ведь к нему могут быть

привязаны платёжные данные. Это реализуется с помощью надежных методов аутентификации, чтобы исключить несанкционированный доступ и предотвратить утечки данных;

– удобство использования. Мобильное приложение должно быть доступно на различных платформах, а именно на Android и iOS. Интуитивно понятный интерфейс и простота использования играют важную роль, позволяя покупателям быстро и без лишних проблем осуществлять заказ.

1.1.1 Обзор аналогов

На сегодняшний день существует большое количество сетевых кафе и крупных заведений, которые представлены по всей России. У них у всех есть своё мобильное приложение, ведь это один из инструментов масштабирования бизнеса. Без этого крупные корпорации могут потерять огромное количество прибыли и самое главное, лояльность потребителя. Малому и среднему бизнесу, который по большей части представлен в одном городе в качестве локальной точки, такое масштабирование не требуется. Также у них нет возможности и средств для запуска крупных маркетинговых компаний, которые требуют мобильного приложения или элементов геймификации.

Наличие собственного сайта у малого и среднего бизнеса в сфере общественного питания – это достаточная редкость. Наличие мобильного приложения – это исключительный случай. Разработать собственное приложение для двух платформ достаточно дорого, поэтому этим и не занимаются.

Для анализа подобных приложений рассмотрим несколько вариантов, таких как Дайкон, Кофейный экспресс и Вертел.

Каждое из этих приложений принадлежит малому бизнесу и предоставляет возможность для заказа блюд. Проведение анализа аналогов поможет выявить особенности и возможности этих приложений, их преимущества и недостатки.

1.1.2 Приложение Дайкон

Дайкон – это уютное заведение, где традиции японской кухни сочетаются с современными тенденциями. Меню включает в себя широкий выбор суши,

					ПГУ1.090301.08.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

роллов, сашими и других восточных деликатесов, приготовленных опытными шеф-поварами.

Мобильное приложение этого заведения оставляет желать лучшего. Интерфейс устарел и перегружен лишними элементами, что затрудняет навигацию. Заказ еды через приложение превращается в настоящее испытание, ведь оно постоянно вылетает. Фотографии блюд низкого качества и часто не соответствуют реальности. Приложение было разработано в 2021 году, но даже на это время его UI выглядит устаревшим и неактуальным, что отрицательно сказывается на общем впечатлении о заведении.

Ниже приведена таблица 1 с плюсами и минусами приложения «Дайкон».

Таблица 1 – Плюсы и минусы приложения «Дайкон»

Плюсы	Минусы
–	Нестабильная работа приложения
–	Устаревший UI
–	Сложная и непонятная навигация

1.1.3 Приложение Кофейный экспресс

Кофейный экспресс – это небольшая камерная кофейня. Здесь посетители могут насладиться разнообразными сортами кофе, разного метода приготовления, и приятной атмосферой. Интерьер заведения выполнен в современном стиле с элементами уюта, благодаря чему посетители могут чувствовать себя как дома.

Мобильное приложение Кофейный экспресс представляет собой каталог напитков, которые представлены в ассортименте кафе. Отсутствует возможность приобретения продукции, приложение осуществляет функцию меню, где можно подробно ознакомиться с каждым напитком. Приложение представлено как на iOS, так и на Android. Одной из главных особенностей является наличие купонной системы, каждый 6-ой кофе отмеченный в приложении – бесплатный.

Ниже приведена таблица 2 с плюсами и минусами приложения «Кофейный экспресс».

Таблица 2 – Плюсы и минусы приложения «Кофейный экспресс»

Плюсы	Минусы
Система купонов	Устаревший UI
Кроссплатформенная совместимость	Отсутствие возможности для заказа
Стабильность работы	—

1.1.4 Приложение Вертел

Шашлычная Вертел – место для любителей мяса, приготовленного на открытом огне. В меню заведения представлены самые разнообразные виды шашлыка: от классического свиного и куриного до экзотических — из баранины и рыбы. Дополняют трапезу свежие овощные салаты, ароматный хлеб и традиционные соусы.

Мобильное приложение Вертела, к сожалению, выполнено не так хорошо, как это заведение. Оно выглядит устаревшим, сложная навигация. Очень большой и трудный путь от выбора блюда до его добавления в корзину. Устаревший UI и отсутствие регулярных обновлений оставляют чувство, что о приложении забыли.

Ниже приведена таблица 3 с плюсами и минусами приложения «Вертел».

Таблица 3 – Плюсы и минусы приложения «Вертел»

Плюсы	Минусы
Информативное меню	Устаревший UI
Стабильность работы	Непонятная навигация и UX

1.1.5 Постановка задачи

Задачей данного дипломного проекта является создание мобильного приложения для кафе под управлением iOS и Android. Приложение должно предоставлять пользователям возможность регистрации, просмотр меню блюд, детальное изучение каждой отдельной позиции и добавление в корзину для осуществления дальнейшего заказа.

Используя данное приложение, пользователь сможет:

- регистрироваться;
- авторизоваться;
- просматривать меню;
- детально изучать отдельные блюда;
- добавлять блюда в корзину.

Диаграмма вариантов использования приложения приведена на рисунке 1.

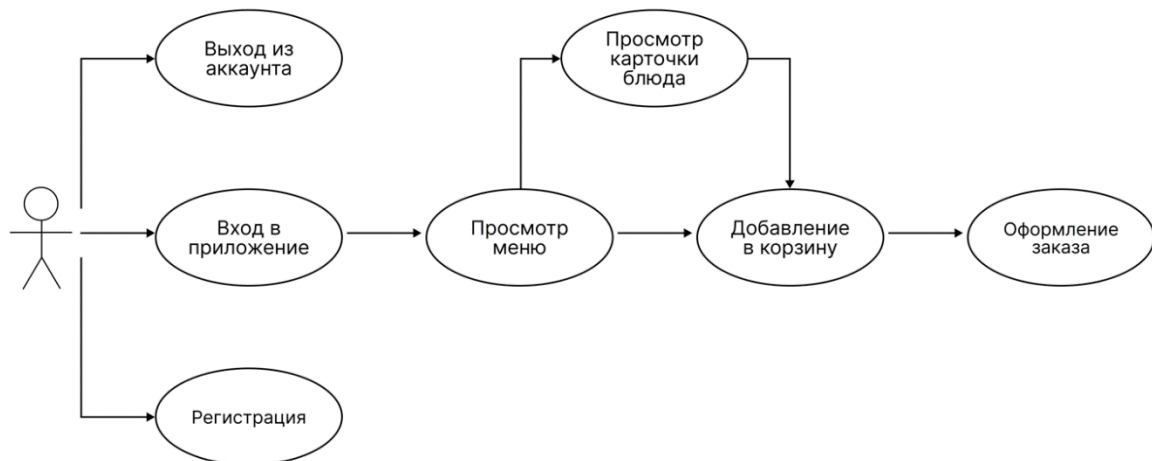


Рисунок 1 – Диаграмма вариантов использования приложения

1.2 Бизнес модель кафе

1.2.1 Общая концепция

Неаполи – это маленькое, классическое пицца-кафе, которое сделано в формате домашней итальянской пиццерии. Это место, где каждый гость может погрузиться в атмосферу Италии и насладиться ее кулинарными традициями. Бизнес ставит перед собой задачу предложить уникальный опыт, который сочетает в себе вкус настоящей итальянской пиццы и домашний уют, ведь большинство подобных аутентичных заведений в Италии принадлежат семьям. Продукт строится на трёх ключевых принципах: качество, аутентичность и внимание к клиенту.

1.2.2 Меню

Меню пицца-кафе «Неаполи» включает в себя только оригинальные пиццы, делающиеся по строгим итальянским рецептам и стандартам:

– Маргарита. Одна из самых известных итальянских пицц с простыми, высококачественными ингредиентами, символизирующими цвета итальянского флага. Ингредиенты: томатный соус, свежая моцарелла, свежий базилик, оливковое масло первого отжима;

– Маринара. Самая старая пицца, которую часто готовят неаполитанские пиццайоло. Простота этой пиццы подчеркивает вкус томатов и аромат чеснока. Ингредиенты: томатный соус, чеснок, орегано, оливковое масло;

– Капричиоза. Ароматная и богатая на вкусы пицца, сочетающая разнообразные ингредиенты. Ингредиенты: томатный соус, моцарелла, артишоки, шампиньоны, ветчина, черные оливки;

– Кватро Стаджони. «Четыре сезона» – каждая четверть этой пиццы представляет другой сезон года со своим уникальным набором ингредиентов. Ингредиенты: томатный соус, моцарелла, артишоки (весна), ветчина (лето), грибы (осень), черные оливки (зима);

– Нордическая. Пицца, вдохновленная северной частью Италии, с добавлением ароматного копченого лосося. Ингредиенты: сметанный соус, моцарелла, копченый лосось, руккола, лимон;

– Фрутти ди Маре. Морепродукты являются основным ингредиентом этой пиццы, которая пользуется популярностью в прибрежных регионах Италии. Ингредиенты: томатный соус, моцарелла, мидии, креветки, кальмары, чеснок, петрушка;

– Пармиджана. Пицца с насыщенным вкусом баклажанов пармиджаны, которая является фаворитом в южной Италии. Ингредиенты: томатный соус, моцарелла, баклажаны, пармезан, базилик;

					ПГУ1.090301.08.001 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

– Дьявола. Подходит для любителей острого. Специи и пикантные колбаски делают эту пиццу поистине «дьявольски» вкусной. Ингредиенты: томатный соус, моцарелла, острая салями, перчики чили, орегано.

Это меню включает в себя самые популярные и аутентичных пиццы Италии, которые можно найти в традиционных итальянских пиццериях.

1.2.3 Цифровые технологии

Введение современных цифровых технологий в работу заведения значительно повышает удобство, комфорт и лояльность клиентов. Мобильное приложение должно позволять легко совершать заказы прямо со смартфона. Через приложение клиенты могут просмотреть полное меню с подробными описаниями блюд и фотографиями, узнать калораж каждой пиццы, зарезервировать столик в пиццерии или совершить заказ непосредственно в кафе.

Программа лояльности, встроенная в приложение, предоставляет пользователям персонализированные предложения и скидки, уведомляет о специальных акциях и автоматически начисляет бонусы за каждый заказ. Также мы внедрили систему отзывов, чтобы клиенты могли делиться своим мнением и оценками, помогая нам постоянно улучшать качество сервиса и продукции.

Исходя из вышеописанной бизнес модели мы можем сделать вывод, что заведение предлагает своим посетителям не только вкусную еду высокого качества, но и уникальный опыт, который объединяет традиции и современные технологии. Каждый гость чувствует себя частью нашей большой итальянской семьи, и это является основой нашего успеха.

1.3 Выбор технологий разработки

1.3.1 Язык программирования Swift

Swift – это высокоуровневый язык программирования, разработанный Apple для создания приложений под iOS, macOS, watchOS и tvOS. Он известен своей скоростью и безопасностью, благодаря строгой типизации и автоматическому управлению памятью (ARC). Swift сочетает в себе функциональные и объектно-

ориентированные парадигмы, что делает его мощным инструментом для разработки. Встроенная поддержка интерфейса предоставляет графическую библиотеку SwiftUI, которая позволяет создавать адаптивные и удобные пользовательские интерфейсы.

1.3.2 Язык программирования Kotlin

Kotlin – это современный статически типизированный язык программирования, разработанный компанией JetBrains, полностью совместимый с Java и запущенными на платформе JVM. Он был создан для повышения производительности и уменьшения количества кода по сравнению с Java. Kotlin поддерживает смешанное программирование (объектно-ориентированное и функциональное), повышая гибкость и выразительность кода. Для создания графических интерфейсов Kotlin может использовать библиотеку Android SDK, а также Jetpack Compose для декларативного программирования интерфейсов.

1.3.3 Фреймворк Flutter

Flutter – это кроссплатформенный фреймворк для создания нативных мобильных, веб-приложений и приложений для настольных систем, разработанный Google. Приложения на Flutter пишутся на языке Dart, который максимально оптимизирован для разработки и отрисовки пользовательских интерфейсов с высокой производительностью. Flutter предоставляет обширный набор виджетов и инструментов для создания сложных и адаптивных пользовательских интерфейсов. Благодаря движку отрисовки Skia, Flutter обеспечивает гибкость и высокую производительность приложений на всех поддерживаемых платформах.

После сравнения возможностей популярных инструментов для создания приложений, был выбран фреймворк Flutter, так как он позволяет разрабатывать кроссплатформенные приложения с единственным кодовым базисом, обеспечивая при этом высокую производительность и богатый набор компонентов интерфейса. Использование виджетов Flutter и библиотеки Dart упрощает создание красивых

пользовательских интерфейсов, которые одинаково хорошо выглядят как на мобильных устройствах, так и на настольных системах.

1.4 Выбор среды разработки

На данный момент в мире информационных технологий существует несколько различных средств разработки мобильных приложений на iOS и Android.

На сегодняшний день самыми востребованными средами являются:

- VS Code;
- Xcode;
- Android Studio.

1.4.1 Visual Studio Code

Эта среда позволяет создавать приложения и программы на самых распространенных языках программирования. Рассмотрим некоторые из достоинств и недостатков Visual Studio Code.

Из достоинств можно отметить кроссплатформенность. Visual Studio Code обеспечивает возможность разработки приложений для различных операционных систем, включая Windows, macOS и Linux, что позволяет разработчикам работать на платформе по своему выбору. Visual Studio Code поддерживает множество расширений, которые позволяют интегрировать различные инструменты, фреймворки и библиотеки, что делает рабочее окружение адаптируемым к различным потребностям и задачам. Visual Studio Code предоставляет широкий спектр инструментов, включая мощный редактор кода с подсветкой синтаксиса, интегрированный терминал, систему контроля версий через Git, отладчик и многое другое, что упрощает и ускоряет процесс разработки.

К недостаткам можно отнести возможные проблемы с производительностью при работе с большими проектами или множеством установленных расширений, что иногда приводит к замедлению работы системы. Также, из-за необходимости установки дополнительных расширений для полного функционала, начальная

конфигурация среды разработки может занимать больше времени по сравнению с другими интегрированными средами. Иногда возникают трудности с интеграцией специфических инструментов и библиотек, что требует дополнительных настроек и может вызвать некоторые недостатки в удобстве работы.

1.4.2 Xcode

Xcode – это интегрированная среда разработки от Apple, предназначенная для создания приложений для iOS, macOS, watchOS и tvOS. Рассмотрим некоторые достоинства и недостатки Xcode.

К достоинствам можно отнести глубокую интеграцию с экосистемой Apple и предоставление всех необходимых инструментов для разработки приложений под устройства Apple. Xcode включает в себя редактор кода с подсветкой синтаксиса, интерфейсный дизайнер, симуляторы устройств и инструмент для контроля версий. Ещё одним преимуществом является возможность использования Swift, нового и мощного языка программирования, разработанного Apple. Xcode поддерживает автоматическое создание и настройку пакетов приложений, что значительно упрощает процесс публикации в App Store. Также стоит отметить высококачественные инструменты для тестирования и профилирования приложений, такие как XCTest и Instruments.

Из недостатков следует отметить ограниченность использования только на macOS, что может быть неудобно для разработчиков, работающих на других операционных системах. Xcode также требует значительных системных ресурсов, и работа с большими проектами может быть медленной на менее мощных компьютерах. Некоторые разработчики могут столкнуться с проблемами стабильности, особенно при использовании бета-версий Xcode. Также процесс обновления Xcode и сопутствующих инструментов может быть сложным и занимает значительное время, что иногда приводит к совместительным проблемам с предыдущими версиями.

1.4.3 Android Studio

Android Studio – это официальная интегрированная среда разработки от Google для создания приложений под платформу Android.

Достоинства. IDE предоставляет множество инструментов для разработки, включая редактор кода с подсветкой синтаксиса, эмуляторы устройств, встроенные инструменты для анализа производительности и отладки. Благодаря тесной интеграции с Android SDK и отличной поддержке Gradle, Android Studio упрощает процесс создания, тестирования и развертывания приложений. Большое сообщество разработчиков поддерживает обширную документацию и множество обучающих ресурсов, а также доступ к множеству плагинов, расширяющих функциональность среды. Android Studio предлагает современные и удобные инструменты для создания пользовательского интерфейса, такие как Layout Editor и Constraint Layout, что позволяет значительно ускорить процесс разработки.

Недостатки. Интерфейс среды может быть сложным для новичков, особенно тех, кто ранее не работал с другими IDE от JetBrains, такими как IntelliJ IDEA. Android Studio требует значительных ресурсов системы, и работа с большими проектами может замедлить производительность на менее мощных компьютерах. Иногда возникают проблемы с Gradle, которые могут затягивать процесс сборки и отладки приложений. Для корректной и удобной работы требуется установка большого количества компонентов, что может занять значительное время и место на диске.

Проанализировав достоинства и недостатки данных сред разработки, было принято решение разрабатывать мобильное приложение с помощью Visual Studio Code в связи с более удобным и интуитивно понятным интерфейсом, большим выбором подключаемых библиотек и возможностью писать приложение в одной IDE для двух платформ.

1.4.4 Firebase

Firebase — это БД от Google, предоставляющая набор облачных сервисов для разработки приложений. Рассмотрим некоторые из достоинств и недостатков Firebase.

Достоинства. Firebase предоставляет широкий спектр инструментов для разработчиков, включая аутентификацию пользователей, базы данных в реальном времени, хранение файлов, анализ использования приложения и многое другое. Одним из ключевых преимуществ является упрощенная интеграция и удобный интерфейс, позволяющий быстро подключать и настраивать различные сервисы. Firebase позволяет синхронизировать данные в реальном времени между клиентами, что особенно полезно для чатов и приложений с активно обновляемым контентом. Платформа также предлагает мощные аналитические инструменты, помогающие отслеживать поведение пользователей и улучшать взаимодействие с приложением. Кроссплатформенная поддержка позволяет разработчикам легко интегрировать Firebase в приложения для iOS, Android и веб.

Недостатки. Firebase имеет ограниченную бесплатную версию с лимитированными ресурсами, которые могут быть недостаточными для крупных проектов. Переход на платные тарифные планы может быть дорогим, особенно для небольших стартапов. Платформа зависит от облачных сервисов Google, что может быть проблемой в случае недоступности или перебоев в работе. В Firebase отсутствует полноценная поддержка SQL-запросов, и разработчики, привыкшие работать с традиционными реляционными базами данных, могут столкнуться с трудностями. Многопоточность и конкурирующий доступ к данным также накладывают некоторые ограничения, особенно в случае высоконагруженных приложений. Наконец, некоторые разработчики могут испытывать трудности с миграцией данных из Firebase на другие платформы или решения, зависимость от которых может стать проблемой в долгосрочной перспективе.

Проанализировав достоинства и недостатки данной базы данных, а также сравнив с её аналогами, было принято решение разрабатывать приложение с помощью Firebase.

1.5 Сравнение ОС

Разработка мобильного приложения для кафе будет происходить под macOS, однако необходимо рассмотреть некоторые операционные системы, чтобы определить их достоинства и недостатки.

Выбор операционной системы играет ключевую роль при создании мобильного приложения, ведь это определяет его платформу. На ОС Windows не получится запустить мобильное iOS-приложение, по причинам отсутствия специализированного симулятора и инструментов разработки, доступных исключительно на macOS. Это значительно усложняет задачу разработчикам, работающим с различными платформами, и требует использования дополнительных ресурсов для запуска и тестирования приложений.

Для Android-приложений ситуация несколько проще, так как Android Studio и эмуляторы Android доступны на разных операционных системах, включая Windows, macOS и Linux. Но и здесь могут возникнуть свои трудности, связанные с настройкой среды разработки и совместимостью с различными версиями операционных систем.

В результате разработчики часто прибегают к использованию кроссплатформенных фреймворков, таких как Flutter, которые позволяют писать код один раз и компилировать его для различных ОС. Это помогает сократить время разработки и тестирования, однако может потребовать дополнительных знаний и навыков работы с этими инструментами.

1.5.1 ОС macOS

Плюсы операционной системы MacOS:

					<i>ПГУ1.090301.08.001 ПЗ</i>	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

– понятный интерфейс. MacOS предлагает простой и удобный для пользователя интерфейс, что делает использование устройств Apple приятным и удобным для людей с любым уровнем технической подготовки;

– высокая безопасность. Система известна своим высоким уровнем безопасности, что защищает личные данные и конфиденциальную информацию от сетевых угроз и утечек;

– взаимодействие с устройствами Apple. Пользователи могут синхронизировать свои данные и продолжать работу на разных устройствах Apple, таких как iPhone, iPad и MacBook;

– богатая база приложений. Пользователь имеет доступ к большому количеству приложений и сервисов, что позволяет выбрать программы для различных целей.

Минусы операционной системы MacOS:

– ограниченная совместимость с оборудованием. Система сильно ограничена в выборе аппаратных средств, так как работает только на устройствах Apple;

– мало игр. MacOS не всегда поддерживает большое количество игр, доступных на других платформах, что ограничивает игровую аудиторию;

– высокие затраты разработки. Стоимость разработки и обновления приложений для macOS высока из-за необходимости использования оборудования Apple и определенной платформы разработки, такой как Xcode;

– узкая пользовательская база. Количество пользователей macOS значительно меньше по сравнению с Windows, что снижает популярность приложений среди пользователей.

1.5.2 ОС Windows

Плюсы операционной системы Windows:

– высокая совместимость. Windows поддерживает множество программ и игр, что делает её популярной среди большого количества пользователей;

– лёгкость в использовании. Интерфейс операционной системы интуитивно понятен для большинства людей, что позволяет быстро освоить и удобно пользоваться её функциями, даже новичкам;

– разнообразие сред разработки. Операционная система совместима с большим количеством устройств и комплектующих, позволяя пользователям выбирать ту среду разработки, которая им наиболее подходит.

Минусы операционной системы Windows:

– дорогие лицензии. Цена лицензий на Windows довольно высокая, особенно для корпоративных клиентов;

– требования к производительности. Некоторые версии Windows требуют более современных и мощных комплектующих для полноценной работы;

– необходимость частых обновлений. Регулярные обновления Windows могут быть неудобными, так как требуют перезагрузки системы, что часто приводит к потере времени и данных.

1.5.3 ОС Linux

Плюсы операционной системы Linux:

– высокий уровень безопасности. Linux славится своей защитой благодаря частым обновлениям и активному сообществу разработчиков;

– хорошие возможности для разработчиков. Несмотря на то, что Linux менее популярен среди пользователей обычных ПК-приложений, он предоставляет широкие возможности для разработчиков;

– разные уровни удобства использования. В зависимости от выбранного дистрибутива, Linux может быть как простым, так и сложным в использовании, требуя от пользователей более глубоких знаний для комфортной работы.

Минусы операционной системы Linux:

– ограниченная поддержка некоторых программ. Хотя Linux поддерживает множество приложений, некоторые программы недоступны для этой платформы.

– разделение на множество дистрибутивов. Большое количество дистрибутивов с разными версиями ядра и библиотек затрудняет разработку и совместимость приложений между различными версиями Linux.

– трудности при установке программ. Для неопытных пользователей процесс установки приложений может быть непривычным и занимать больше времени.

1.6 Проектирование мобильного приложения

1.6.1 Архитектура приложения

Архитектура Flutter BloC – это не совсем архитектура, а так называемый State Manager, средство по управлению состояниями. В отличие от архитектуры MVVM, где у нас данные хранятся в модели, логика обрабатывается во ViewModel, а отображается во View. В архитектуре Flutter BLoC всё сделано по-другому. Здесь данные хранятся не в моделях, а в сущностях – Entities. Все методы по запросу данных с сервера должны находиться в репозиториях –Repository. Все возможные состояния бизнес-логики хранятся в классе State, события изменения состояний – в Event. Класс Bloc же выступает в роли той самой ViewModel: вызывает методы из репозитория, и в зависимости от их ответа меняет State. В самом View, который во Flutter называется Widget, то есть экран, мы показываем определенные UI-элементы в зависимости от текущего State, не зная, что происходит «под капотом». В этом и вся суть – мы максимально разделяем бизнес-логику и UI, всё по принципам SOLID. Пример подобной архитектуры приведен на рисунке 2.

Всё взаимодействие пользователя будет осуществляться через мобильное приложение. Приложение состоит из двух ключевых компонентов:

- клиентская часть, которая обеспечивает пользовательский интерфейс для взаимодействия с системой, включая кнопки, поля и др.;
- база данных, где хранятся данные с определенными правилами и принципами управления данными.

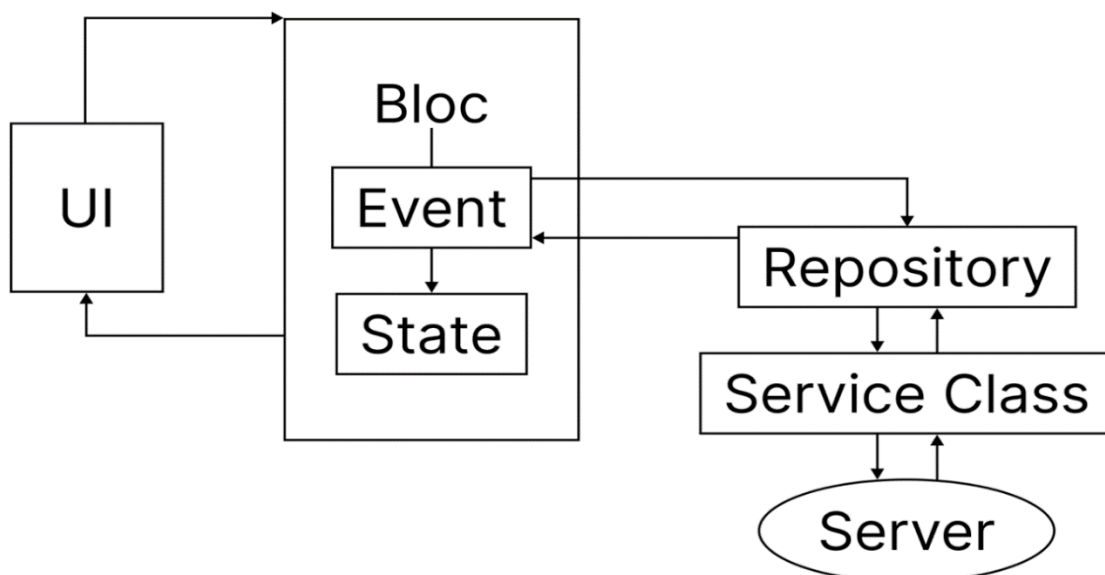


Рисунок 2 – Архитектура Flutter BloC

Firebase представляет собой платформу с набором облачных инструментов, которые позволяют разработчикам создавать, управлять и масштабировать приложения. Она предоставляет возможности для организации, контроля и администрирования данных в режиме реального времени, обеспечивая синхронизацию данных между клиентами и сервером без необходимости создания собственного сервера или управления инфраструктурой. Это изображено на рисунке 3.

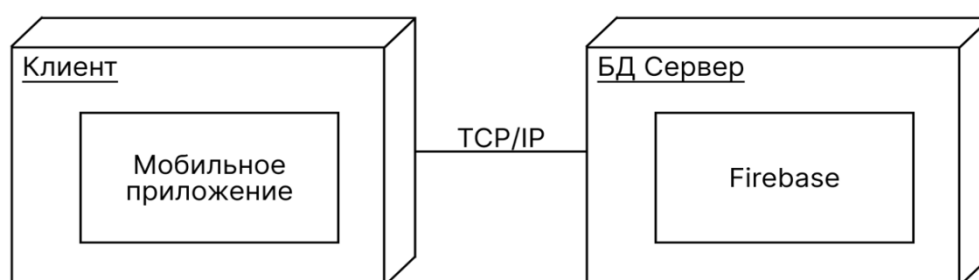


Рисунок 3 – Диаграмма развёртывания приложения

На рисунке 4 представлено взаимодействие компонентов приложения в виде диаграммы последовательности.

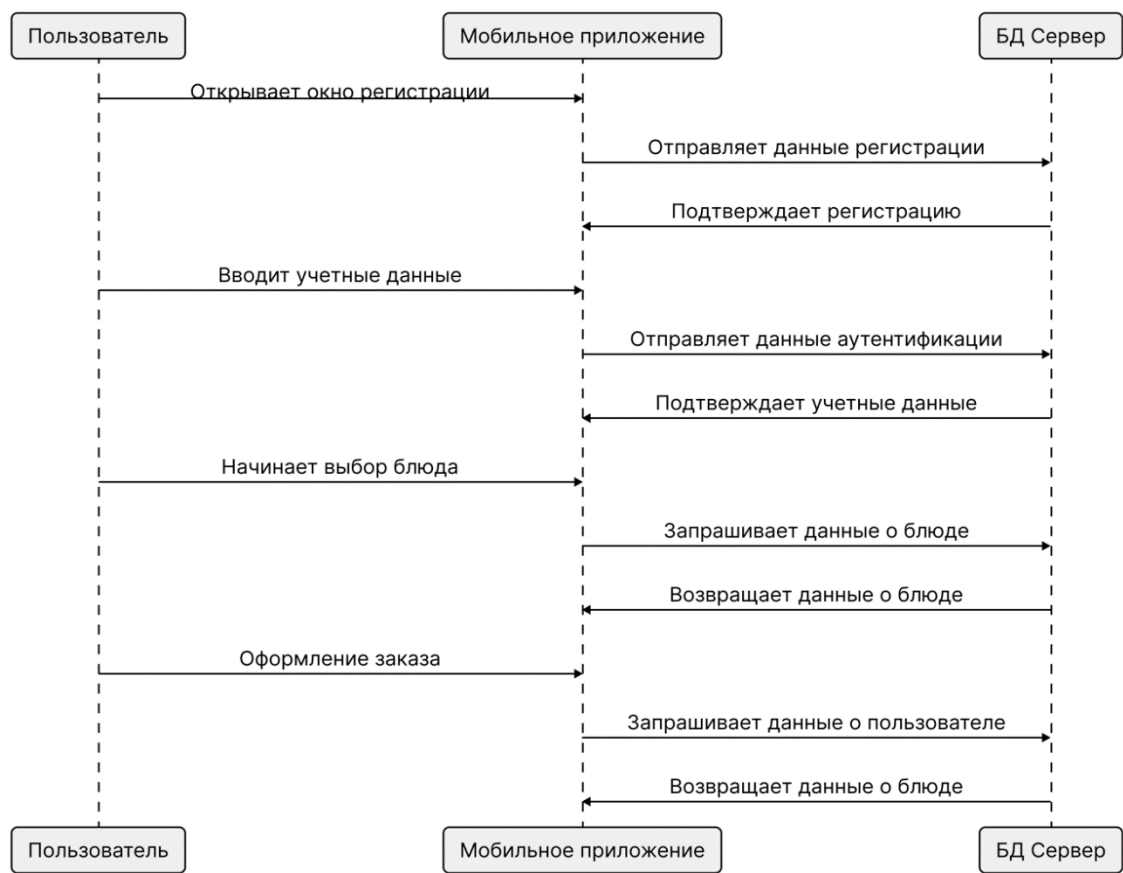


Рисунок 4 – Диаграмма последовательности

На UML-диаграмме классов можно увидеть основные файлы проекта (рисунок 5).

Каждый из них отвечает за свою задачу, что соответствует принципу разделения ответственности и модульности:

В файле `main.dart` вызывается основная глобальная функция `main()`, которая инициализирует ключевые компоненты приложения — `Firebase` и `Bloc`, а также запускает главный экран приложения.

В классе MyAppView задается начальный экран приложения: если пользователь зарегистрирован, происходит переход к каталогу товаров, иначе — на приветственный экран.

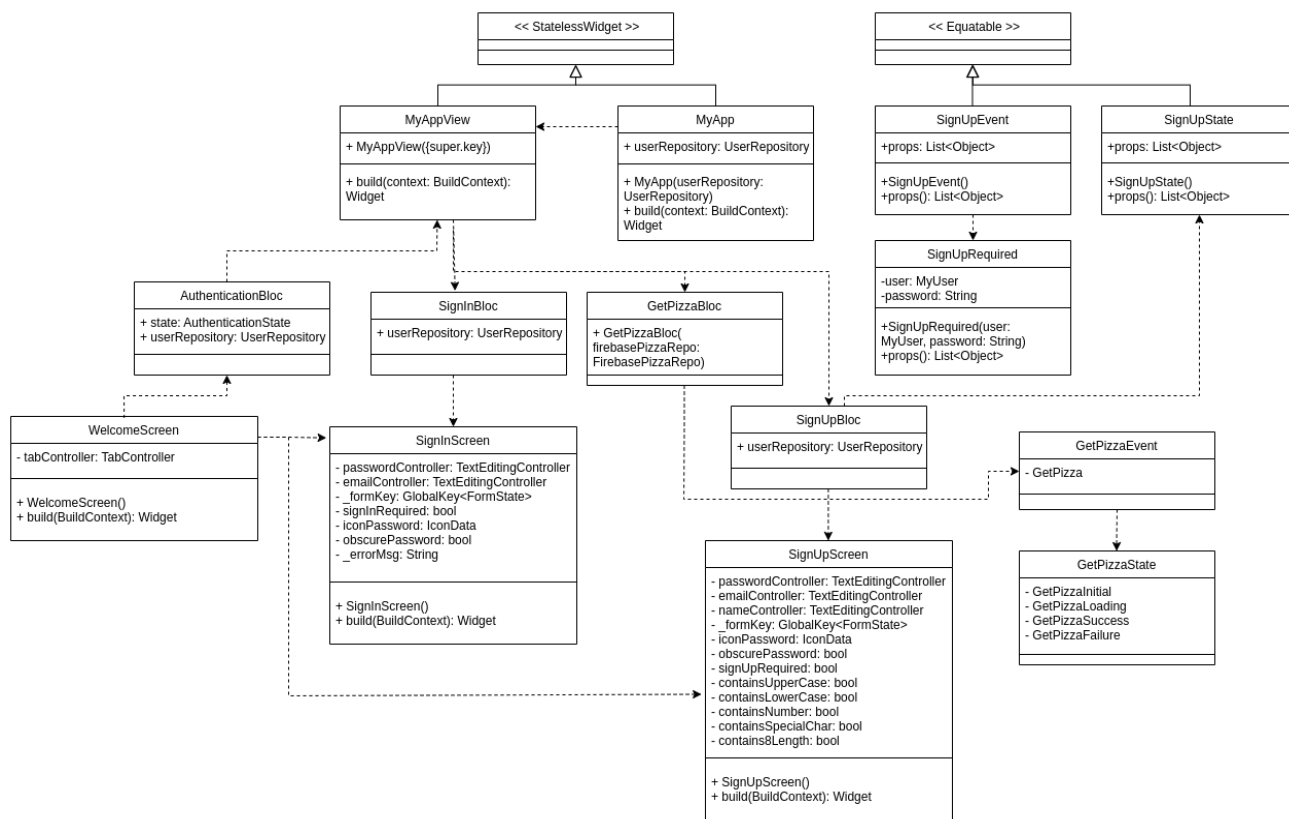


Рисунок 5 – UML-диаграмма классов

В классе AuthenticationBloc осуществляется проверка состояний пользователя и подписка на их изменения. В классах SignUpBloc и SignInBloc реализована логика вызова методов регистрации и авторизации соответственно. В зависимости от успешности ответа происходит изменение состояний SignUpState и SignInState, которые, в свою очередь, влияют на итоговое состояние AuthenticationState.

Класс SignUpScreen отвечает за отображение формы регистрации, класс SignInScreen — за отображение формы входа, а класс WelcomeScreen демонстрирует экран приветствия.

Класс GetPizzaBloc отвечает за бизнес-логику, связанную с пиццей. Он вызывает метод getPizzas из PizzaRepository и в зависимости от его успеха

изменяет свое состояние GetPizzaState. В классе HomeScreen отображается список всех полученных пицц, а в классе DetailsScreen – подробная информация о каждой пицце.

1.6.2 Проектирование базы данных

Для хранения информации необходимо реализовать базу данных со следующими таблицами:

– users – таблица, содержащая данные учетных записей клиентво заведения (таблица 4);

– pizzas – таблица, содержащая данные блюд заведения (таблица 5).

Таблица 4 – Атрибуты таблицы users

Имя столбца	Тип данных	Описание	Пример
userId	STRING	Идентификатор учетной записи	jv93VuefDdfSNBnfaa mfdp4FM2R2
email	STRING	Email для входа	bogdankireev@mail.ru
name	STRING	Имя для регистрации	Богдан

Таблица 5 – Атрибуты таблицы pizzas

Имя столбца	Тип данных	Описание	Пример
pizzaId	STRING	Идентификатор блюда	aOnQezTjbAkDyam1lCB1
name	STRING	Название блюда	Сырная
description	STRING	Описание блюда	Моцарелла, пармезан, фирменный соус альфредо
price	NUMBER	Цена блюда	999
discount	NUMBER	Цена блюда с учетом скидки	800
spicy	NUMBER	Идентификатор острого блюда	1
isVeg	BOOLEAN	Идентификатор блюда без мяса	true

Имя столбца	Тип данных	Описание	Пример
pizzaId	STRING	Идентификатор блюда	aOnQezTjbAkDyam1lCBl
name	STRING	Название блюда	Сырная
description	STRING	Описание блюда	Моцарелла, пармезан, фирменный соус альфредо
price	NUMBER	Цена блюда	999
discount	NUMBER	Цена блюда с учетом скидки	800
picture	STRING	Ссылка на фотографию блюда	https://firebasestorage.googleapis.com/v0/b/pizza-project-ba0b5.appspot.com/o/1.png?alt=media&token=d51a8bde-31bd-4c44-b915-489a8a0b53af
macros	MAP (NUMBER)	Данные о калораже блюд	calories: 267 carbs: 38 fat: 21

1.7 Реализация приложения

1.7.1 Реализация регистрации и аутентификации пользователя

Была создана система аутентификации пользователей. Аутентификация использует класс WelcomeScreen. Пользователю необходимо ввести email и пароль в соответствующие поля. Это проиллюстрировано на рисунке 6.

Код программы представлен ниже.

```
class WelcomeScreen extends StatefulWidget {
  const WelcomeScreen({super.key});

  @override
  State<WelcomeScreen> createState() => _WelcomeScreenState();
}

class _WelcomeScreenState extends State<WelcomeScreen>
  with TickerProviderStateMixin {
  late TabController tabController;
  @override
  void initState() {
    tabController = TabController(initialIndex: 0, length: 2,
vsync: this);
    super.initState();
  }
}
```

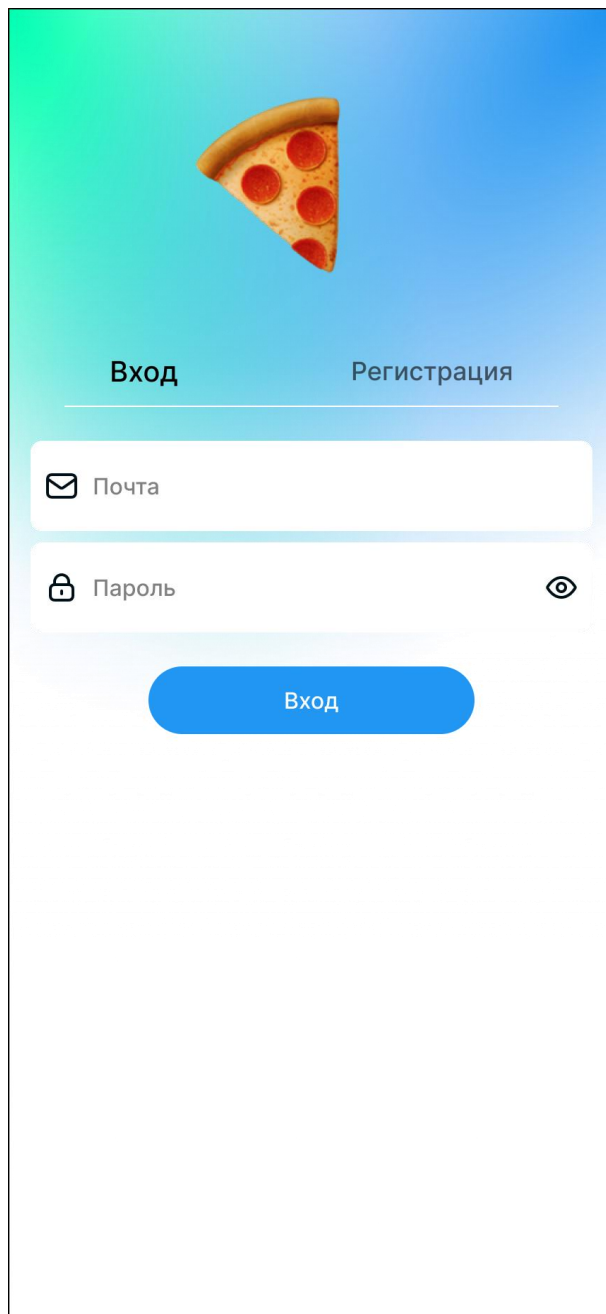


Рисунок 6 – Экран входа в приложение

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor:
Theme.of(context).colorScheme.background,
    body: SingleChildScrollView(
      child: SizedBox(
        height: MediaQuery.of(context).size.height,
        child: Stack(
          children: [
            Align(
```

```

1.2),
        alignment: const AlignmentDirectional(20, -
        child: Container(
            height: MediaQuery.of(context).size.width,
            width: MediaQuery.of(context).size.width,
            decoration: BoxDecoration(
                shape: BoxShape.circle,
                color:
Theme.of(context).colorScheme.tertiary),
        ),
    ),
    Align(
        alignment: const AlignmentDirectional(2.7, -
1.2),
        child: Container(
            height: MediaQuery.of(context).size.width /
1.3,
            width: MediaQuery.of(context).size.width /
1.3,
            decoration: BoxDecoration(
                shape: BoxShape.circle,
                color:
Theme.of(context).colorScheme.primary),
        ),
    ),
    BackdropFilter(
        filter: ImageFilter.blur(sigmaX: 100.0, sigmaY:
100.0),
        child: Container(),
    ),
    Align(
        alignment: Alignment.center,
        child: SizedBox(
            height: MediaQuery.of(context).size.height /
1.8,
            child: Column(
                children: [
                    Padding(
                        padding: const
EdgeInsets.symmetric(horizontal: 50.0),
                        child: TabBar(
                            controller: tabController,
                            unselectedLabelColor:
Theme.of(context)
                                .colorScheme
                                .onBackground
                                .withOpacity(0.5),
                            labelColor:
Theme.of(context).colorScheme.onBackground,
                            tabs: const [
                                Padding(

```


Если пользователь ещё не зарегистрирован в системе, то ему необходимо зарегистрироваться. Для этого необходимо ввести следующие данные:

- электронная почта;
- пароль;
- имя.

Важное условие, чтобы пароль удовлетворял нижепредставленным требованиям:

- 1 символ верхнего регистра;
- 1 символ нижнего регистра;
- 1 цифра;
- 1 специальный символ;
- минимум 8 символов.

Экран регистрации в приложении приведен на рисунке 7.

Код программы представлен ниже.

```
class SignUpScreen extends StatefulWidget {  
  const SignUpScreen({super.key});  
  
  @override  
  State<SignUpScreen> createState() => _SignUpScreenState();  
}  
  
class _SignUpScreenState extends State<SignUpScreen> {  
  final passwordController = TextEditingController();  
  final emailController = TextEditingController();  
  final nameController = TextEditingController();  
  final _formKey = GlobalKey<FormState>();  
  IconData iconPassword = CupertinoIcons.eye_fill;  
  bool obscurePassword = true;  
  bool signUpRequired = false;  
  bool containsUpperCase = false;  
  bool containsLowerCase = false;  
  bool containsNumber = false;  
  bool containsSpecialChar = false;  
  bool contains8Length = false;
```

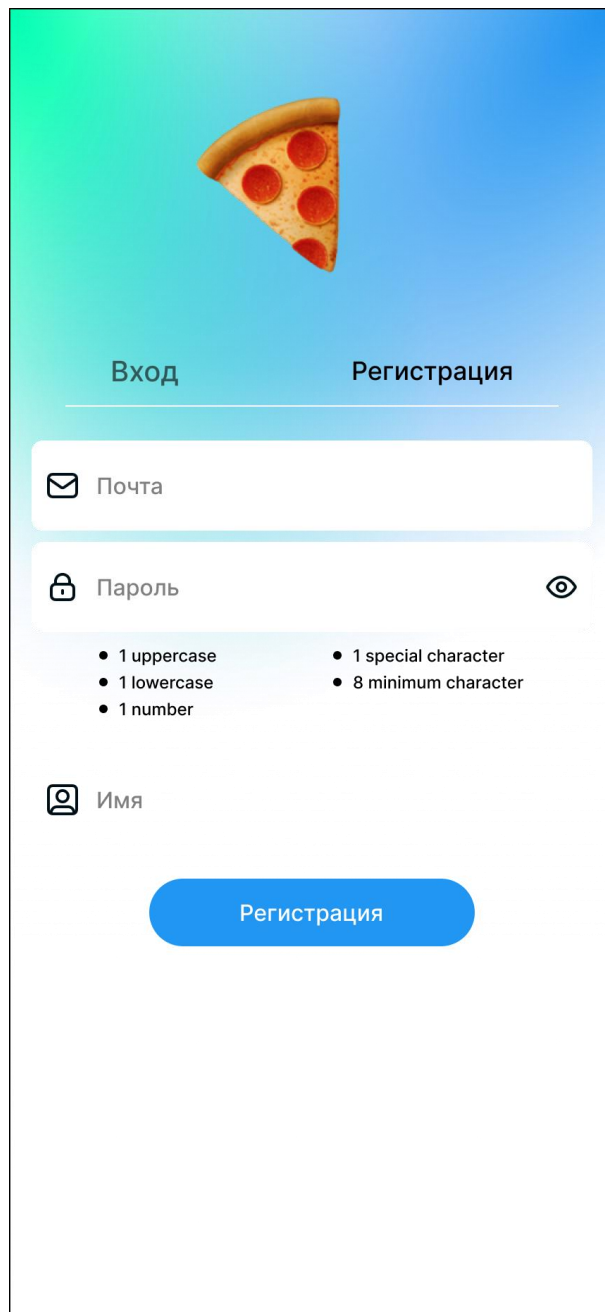


Рисунок 7 – Экран регистрации в приложении

```
@override
Widget build(BuildContext context) {
  return BlocListener<SignUpBloc, SignUpState>(
    listener: (context, state) {
      if (state is SignUpSuccess) {
        setState(() {
          signUpRequired = false;
        });
      } else if (state is SignUpProcess) {
        setState(() {
          signUpRequired = true;
        });
      }
    }
  );
}
```

```

        } else if (state is SignUpFailure) {
            return;
        }
    },
    child: Form(
        key: _formKey,
        child: Center(
            child: Column(
                children: [
                    const SizedBox(height: 20),
                    SizedBox(
                        width: MediaQuery.of(context).size.width * 0.9,
                        child: MyTextField(
                            controller: emailController,
                            hintText: 'Почта',
                            obscureText: false,
                            keyboardType: TextInputType.emailAddress,
                            prefixIcon: const
Icon(CupertinoIcons.mail_solid),
                            validator: (val) {
                                if (val!.isEmpty) {
                                    return 'Пожалуйста введите поле';
                                } else if (!RegExp(r'^[\w-\.]+' + @([\w-
]+.)+[\w-]{2,4}$'))
                                    .hasMatch(val)) {
                                    return 'Пожалуйста введите правильный
email';
                                }
                                return null;
                            }
                        ),
                    const SizedBox(height: 10),
                    SizedBox(
                        width: MediaQuery.of(context).size.width * 0.9,
                        child: MyTextField(
                            controller: passwordController,
                            hintText: 'Пароль',
                            obscureText: obscurePassword,
                            keyboardType:
TextInputType.visiblePassword,
                            prefixIcon: const
Icon(CupertinoIcons.lock_fill),
                            onChanged: (val) {
                                if (val!.contains(RegExp(r'[A-Z]')) {
                                    setState(() {
                                        containsUpperCase = true;
                                    });
                                } else {
                                    setState(() {
                                        containsUpperCase = false;
                                    });
                                }
                            }
                        )
                    )
                ]
            )
        )
    )

```

```

        if (val.contains(RegExp(r'[a-z]')))) {
            setState(() {
                containsLowerCase = true;
            });
        } else {
            setState(() {
                containsLowerCase = false;
            });
        }
        if (val.contains(RegExp(r'[0-9]')))) {
            setState(() {
                containsNumber = true;
            });
        } else {
            setState(() {
                containsNumber = false;
            });
        }
        if (val.contains(RegExp(
            r'^(?=.*?[@#$$&*~`)\%\-
(_+=;:,.<>/?"[{\}\|^\]')')))) {
            setState(() {
                containsSpecialChar = true;
            });
        } else {
            setState(() {
                containsSpecialChar = false;
            });
        }
        if (val.length >= 8) {
            setState(() {
                contains8Length = true;
            });
        } else {
            setState(() {
                contains8Length = false;
            });
        }
        return null;
    },
    suffixIcon: IconButton(
        onPressed: () {
            setState(() {
                obscurePassword = !obscurePassword;
                if (obscurePassword) {
                    iconPassword =
CupertinoIcons.eye_fill;
                } else {
                    iconPassword =
CupertinoIcons.eye_slash_fill;
                }
            });
        }
    );

```

```

        },
        icon: Icon(iconPassword),
    ),
    validator: (val) {
        if (val!.isEmpty) {
            return 'Пожалуйста введите поле';
        } else if (!RegExp(
            r'^(?=.*?[A-Z])(?=.*?[a-
z])(?=.*?[0-9])(?=.*?[!@#\$&*~`)\%\-(_+=;:,.<>/?"[{\\}]|^|^\^)).{8,}$')
            .hasMatch(val)) {
            return 'Пожалуйста введите правильный
пароль';
        }
        return null;
    })),
    const SizedBox(height: 10),
    Row(
        mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
            Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                    Text(
                        "● 1 uppercase",
                        style: TextStyle(
                            color: containsUpperCase
                                ? Colors.green
                                :
Theme.of(context).colorScheme.onBackground),
                    ),
                    Text(
                        "● 1 lowercase",
                        style: TextStyle(
                            color: containsLowerCase
                                ? Colors.green
                                :
Theme.of(context).colorScheme.onBackground),
                    ),
                    Text(
                        "● 1 number",
                        style: TextStyle(
                            color: containsNumber
                                ? Colors.green
                                :
Theme.of(context).colorScheme.onBackground),
                    ),
                ],
            ),
        ],
    ),

```

```

        ),
        Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
                Text(
                    "● 1 special character",
                    style: TextStyle(
                        color: containsSpecialChar
                            ? Colors.green
                            :
Theme.of(context).colorScheme.onBackground),
                ),
                Text(
                    "● 8 minimum character",
                    style: TextStyle(
                        color: contains8Length
                            ? Colors.green
                            :
Theme.of(context).colorScheme.onBackground),
                ),
            ],
        ),
        ],
    ),
    const SizedBox(height: 10),
    SizedBox(
        width: MediaQuery.of(context).size.width * 0.9,
        child: MyTextField(
            controller: nameController,
            hintText: 'Имя',
            obscureText: false,
            keyboardType: TextInputType.name,
            prefixIcon: const
Icon(CupertinoIcons.person_fill),
            validator: (val) {
                if (val!.isEmpty) {
                    return 'Пожалуйста заполните поле';
                } else if (val.length > 30) {
                    return 'Имя слишком длинное';
                }
                return null;
            }
        ),
        SizedBox(height:
MediaQuery.of(context).size.height * 0.02),
        !signupRequired
            ? SizedBox(
                width: MediaQuery.of(context).size.width
* 0.5,
                child: TextButton(
                    onPressed: () {

```

```

        if
(_formKey.currentState!.validate()) {
            MyUser myUser = MyUser.empty;
            myUser.email =
emailController.text;
            myUser.name =
nameController.text;

            setState(() {

context.read<SignUpBloc>().add(SignUpRequired(
                myUser,
passwordController.text));
            });
        },
        style: TextButton.styleFrom(
            elevation: 3.0,
            backgroundColor:

Theme.of(context).colorScheme.primary,
            foregroundColor: Colors.white,
            shape: RoundedRectangleBorder(
                borderRadius:

BorderRadius.circular(60))),
        child: const Padding(
            padding: EdgeInsets.symmetric(
                horizontal: 25, vertical: 5),
            child: Text(
                'Регистрация',
                textAlign: TextAlign.center,
                style: TextStyle(
                    color: Colors.white,
                    fontSize: 16,
                    fontWeight: FontWeight.w600),
            ),
        )),
        )
        : const CircularProgressIndicator()
    ],
    ),
    ),
    ),
);
}
}

```

Данные электронной почты и имя пользователя отправляются в облачную базу данных Firebase. Это приведено на рисунке 8.

					ПГУ1.090301.08.001 ПЗ	Лист
						38
Изм.	Лист	№ докум.	Подпись	Дата		

Search by email address, phone number, or user UID

Add user

Identifier	Providers	Created ↓	Signed In	User UID
bogdankireev@mail.ru		Jun 7, 2024	Jun 7, 2024	jv93VuefDdfSNBnfaamfdp4F...
test@mail.ru		May 15, 2024	Jun 17, 2024	LNsxFbcXrET0GR1m35MVXW...

Rows per page: 50 1 – 2 of 2

Рисунок 8 – Входные данные в облачной БД Firebase

1.7.2 Реализация меню

После успешного входа в приложение, вызывается бла бла бла, которое открывает экран меню. Меню состоит из набора карточек блюд, где указаны следующие данные:

- название;
- описание;
- цена;
- цена со скидкой.

Экран реализации меню переведен на рисунке 9.

Вышеописанные данные представлены в облачной базе данных Firebase, откуда они и «подтягиваются» в приложение. Это отображено на рисунке 10.

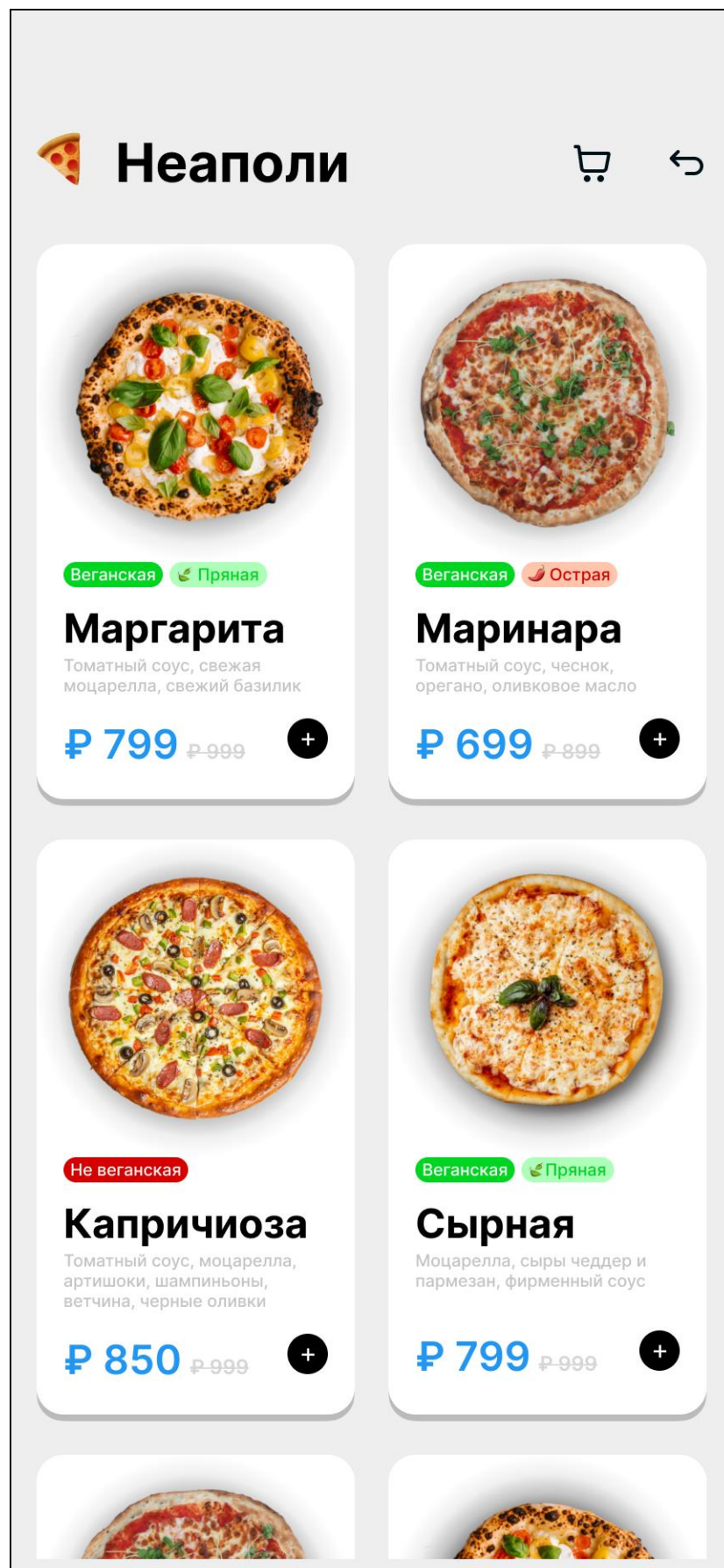


Рисунок 9 – Экран реализации меню

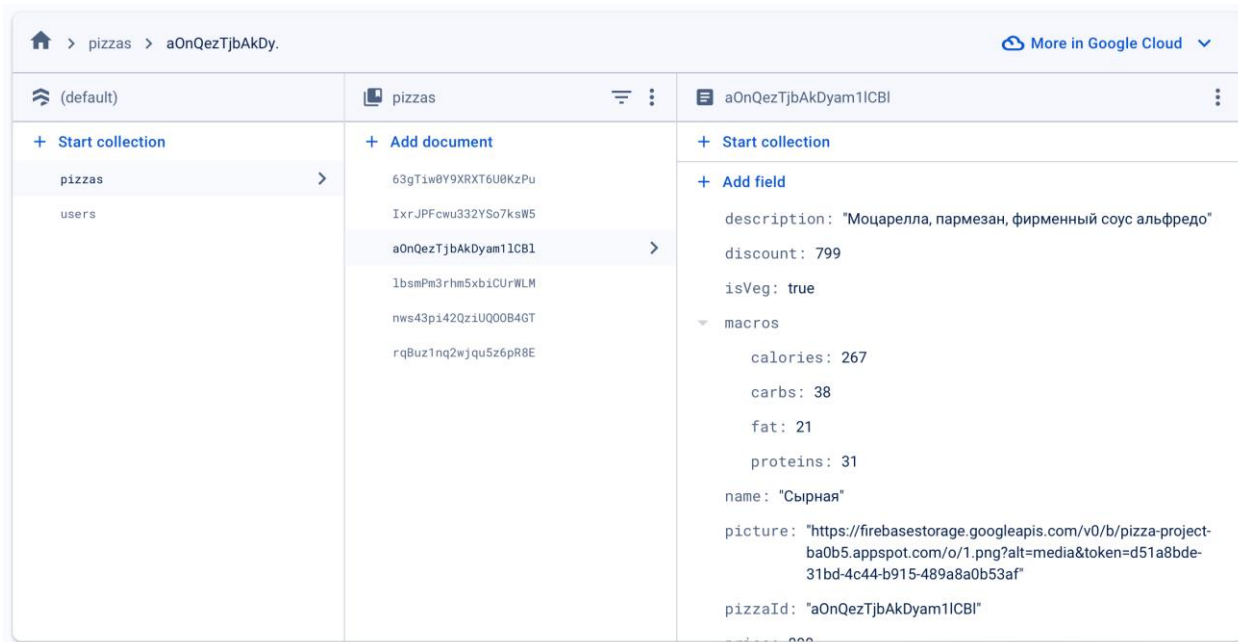


Рисунок 10 – Данные о блюде в БД Firebase

Все изображения также находятся в Firebase, а именно во вкладке «Storage». В базе данных указана url-ссылка на каждое отдельное изображение в хранилище (рисунок 11).

gs://pizza-project-ba0b5.appspot.com					Upload file		
☐	Name	Size	Type	Last modified			
☐	1.png	346.66 KB	image/png	Jun 6, 2024			
☐	2.png	393.65 KB	image/png	Jun 6, 2024			
☐	3.png	365.64 KB	image/png	Jun 6, 2024			
☐	4.png	402.97 KB	image/png	Jun 6, 2024			
☐	5.png	366.59 KB	image/png	Jun 6, 2024			
☐	6.png	395.42 KB	image/png	Jun 6, 2024			

Рисунок 11 – «Storage» в БД Firebase

Код программы представлен ниже.

```

class HomeScreen extends StatelessWidget {
  const HomeScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(

```

```

        backgroundColor:
Theme.of(context).colorScheme.background,
        appBar: AppBar(
          backgroundColor:
Theme.of(context).colorScheme.background,
          title: Row(
            children: [
              Image.asset('assets/8.png', scale: 14),
              const SizedBox(width: 8),
              const Text(
                'Неаполи',
                style: TextStyle(fontWeight: FontWeight.w900,
fontSize: 30),
              )
            ],
          ),
          actions: [
            IconButton(onPressed: () {}, icon: const
Icon(CupertinoIcons.cart)),
            IconButton(
              onPressed: () {

context.read<SignInBloc>().add(SignOutRequired());
              },
              icon: const
Icon(CupertinoIcons.arrow_right_to_line)),
          ],
          body: Padding(
            padding: const EdgeInsets.all(16.0),
            child: BlocBuilder<GetPizzaBloc, GetPizzaState>(
              builder: (context, state) {
                if (state is GetPizzaSuccess) {
                  return GridView.builder(
                    gridDelegate: const
SliverGridDelegateWithFixedCrossAxisCount(
                      crossAxisCount: 2,
                      crossAxisSpacing: 16,
                      mainAxisSpacing: 16,
                      childAspectRatio: 9 / 16),
                    itemCount: state.pizzas.length,
                    itemBuilder: (context, int i) {
                      return Material(
                        elevation: 3,
                        color: Colors.white,
                        shape: RoundedRectangleBorder(
                          borderRadius:
BorderRadius.circular(20),
                        ),
                        child: InkWell(
                          borderRadius:
BorderRadius.circular(20),

```

```

onTap: () {
  Navigator.push(
    context,
    MaterialPageRoute<void>(
      builder: (BuildContext context)
=>
DetailsScreen(state.pizzas[i]),
    ),
  );
},
child: Column(
  crossAxisAlignment:
CrossAxisAlignment.start,
  children: [
Image.network(state.pizzas[i].picture),
    Padding(
      padding:
        const
EdgeInsets.symmetric(horizontal: 12.0),
      child: Row(
        children: [
          Container(
            decoration: BoxDecoration(
              color:
state.pizzas[i].isVeg
                ? Colors.green
                : Colors.red,
              borderRadius:
BorderRadius.circular(30)),
            child: Padding(
              padding: const
EdgeInsets.symmetric(
                vertical: 4,
                horizontal: 8),
              child: Text(
                state.pizzas[i].isVeg
                  ? "Веганская"
                  : "Мясная",
                style: const TextStyle(
                  color:
Colors.white,
                  fontWeight:
FontWeight.bold,
                  fontSize: 10),
                ),
              ),
            ),
            const SizedBox(width: 8),
            Container(

```

					ПГУ1.090301.08.001 ПЗ	Лист
						43
Изм.	Лист	№ докум.	Подпись	Дата		

```

Colors.green.withOpacity(0.2),
BorderRadius.circular(30)),
EdgeInsets.symmetric(
horizontal: 8),
== 1
state.pizzas[i].spicy == 2
state.pizzas[i].spicy == 1
state.pizzas[i].spicy == 2
Colors.orange
Colors.redAccent,
FontWeight.w800,
decoration: BoxDecoration(
  color:
    borderRadius:
child: Padding(
  padding: const
    vertical: 4,
child: Text(
  state.pizzas[i].spicy
    ? "🔥 Прямая"
    :
      ? "🔥 Баланс"
      : "🔥 Острая",
style: TextStyle(
  color:
    ? Colors.green
    :
      ?
      :
fontWeight:
fontSize: 10),
),
),
),
),
),
const SizedBox(height: 8),
Padding(
  padding:
    const
EdgeInsets.symmetric(horizontal: 12.0),
  child: Text(
    state.pizzas[i].name,
    style: const TextStyle(
      fontSize: 20, fontWeight:
        FontWeight.bold),
  ),
),
Padding(

```

```

padding:
  const
EdgeInsets.symmetric(horizontal: 12.0),
  child: Text(
    state.pizzas[i].description,
    style: TextStyle(
      fontSize: 10,
      color: Colors.grey.shade500,
    ),
  ),
),
Padding(
  padding: const
EdgeInsets.symmetric(
    horizontal: 12.0),
  child: Row(
    mainAxisAlignment:
MainAxisAlignment.spaceBetween,
    children: [
      Row(
        children: [
          Text(
            "\P
${state.pizzas[i].discount}",
            style: TextStyle(
              fontSize: 16,
              color:
                .colorScheme
                .primary,
              fontWeight:
FontWeight.w700),
          const SizedBox(
            width: 5,
          ),
          Text(
            "\P
${state.pizzas[i].price}",
            style: TextStyle(
              fontSize: 12,
              color:
                Colors.grey.shade500,
              fontWeight:
FontWeight.w700,
              decoration:
TextDecoration.lineThrough),
          ),
        ],
      ),
    ],
  ),
),

```

					ПГУ1.090301.08.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

```

        IconButton(
            onPressed: () {},
            icon: const Icon(
CupertinoIcons.add_circled_solid))
        ],
    ),
),
);
});
} else if (state is GetPizzaLoading) {
    return const Center(
        child: CircularProgressIndicator(),
    );
} else {
    return const Center(
        child: Text("An error has occurred..."),
    );
}
},
),
),
);
}
}

```

1.7.3 Реализация карточки блюда

При нажатии на карточку блюда, открывается отдельный экран, в котором представлена подробная информация о еде, а именно:

- название;
- цена;
- цена со скидкой;
- описание;
- калораж;
- белки;
- жиры;
- углеводы.

Это отображено на рисунке 12.

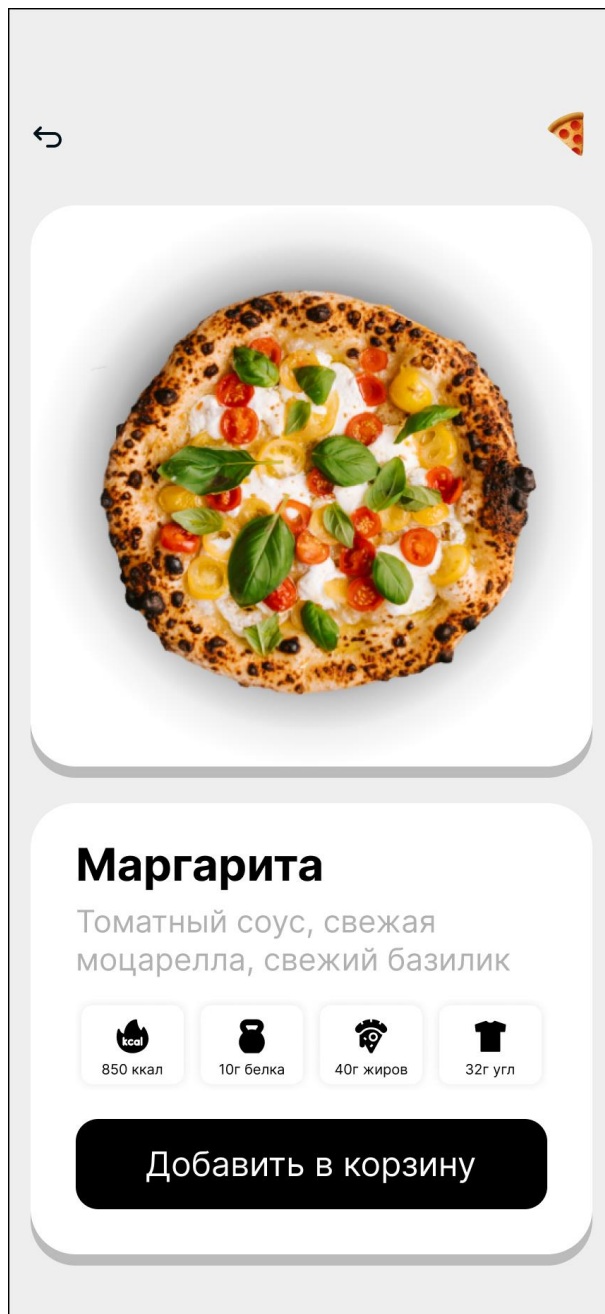


Рисунок 12 – Подробная карточка блюда

Код программы представлен ниже.

```
class DetailsScreen extends StatelessWidget {
  final Pizza pizza;
  const DetailsScreen(this.pizza, {super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor:
        Theme.of(context).colorScheme.background,
```

[illegible]

					<div style="text-align: center;"> <h1>ПГУ1.090301.08.001 ПЗ</h1> </div>	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

```

Expanded(
  flex: 1,
  child: Align(
    alignment: Alignment.centerRight,
    child: Column(
      crossAxisAlignment:
CrossAxisAlignment.end,

      children: [
        Text(
          "\P ${pizza.discount}",
          style: TextStyle(
            fontSize: 20,
            fontWeight:
FontWeight.bold,

            color: Theme.of(context)
              .colorScheme
              .primary),
        ),
        Text(
          "\P ${pizza.price}",
          style: const TextStyle(
            fontSize: 16,
            fontWeight:
FontWeight.bold,

            color: Colors.grey,
            decoration:
TextDecoration.lineThrough),
        ),
      ],
    ),
  ),
),
const SizedBox(
  height: 12,
),
Row(
  children: [
    MyMacroWidget(
      title: "ККал",
      value: pizza.macros.calories,
      icon: FontAwesomeIcons.fire,
    ),
    const SizedBox(
      width: 10,
    ),
    MyMacroWidget(
      title: "Белка",
      value: pizza.macros.proteins,
      icon: FontAwesomeIcons.dumbbell,
    ),
  ],
),

```

```
BorderRadius.circular(10))),
```

ПГУ1.090301.08.001 ПЗ

1.7.4 Реализация корзины

Экран корзины представляет собой главное место для управления покупками пользователя. Он содержит:

- список товаров: перечень всех добавленных в корзину товаров с их изображениями, названиями и ценами;
- общая сумма: в нижней части экрана отображена общая сумма всех товаров в корзине;
- кнопка оформления заказа: под списком товаров расположена крупная и заметная кнопка, которая позволяет пользователю перейти к процессу оплаты.

Это отображено на рисунке 13.

Код программы представлен ниже.

```
class DetailsScreen extends StatelessWidget {
  final Pizza pizza;
  const DetailsScreen(this.pizza, {super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor:
Theme.of(context).colorScheme.background,
      appBar: AppBar(
        backgroundColor:
Theme.of(context).colorScheme.background,
      ),
      body: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 20),
        child: Column(
          children: [
            Container(
              width: MediaQuery.of(context).size.width,
              height: MediaQuery.of(context).size.width - (40),
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.circular(30),
                boxShadow: const [
                  BoxShadow(
                    color: Colors.grey, offset: Offset(3,
3), blurRadius: 5)
                ],
              image: DecorationImage(image:
NetworkImage(pizza.picture))),
          ],
        ),
      ),
    );
  }
}
```

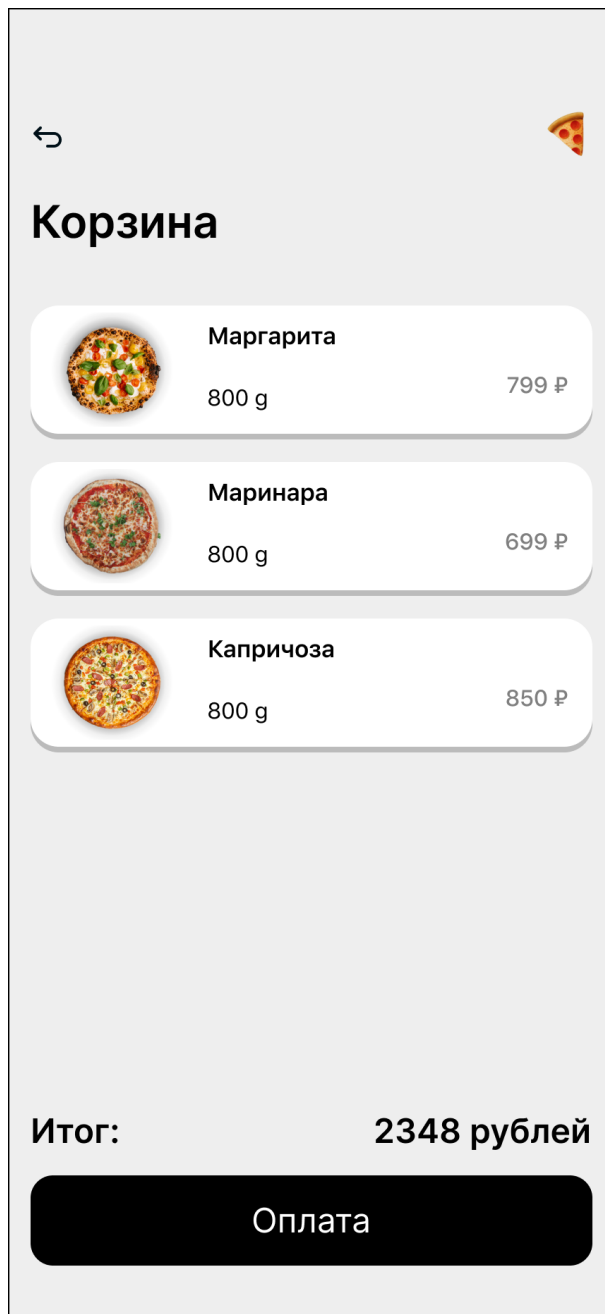


Рисунок 13 – Реализация экрана корзины

```
const SizedBox(
  height: 30,
),
Container(
  decoration: BoxDecoration(
    color: Colors.white,
    borderRadius: BorderRadius.circular(30),
    boxShadow: const [
      BoxShadow(
        color: Colors.grey, offset: Offset(3, 3),
        blurRadius: 5)
    ],
  ),
```

```

),
child: Padding(
  padding: const EdgeInsets.all(20.0),
  child: Column(
    children: [
      Row(
        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
        children: [
          Expanded(
            flex: 2,
            child: Text(
              pizza.name,
              style: const TextStyle(
                fontSize: 20, fontWeight:
FontWeight.bold),
            ),
          ),
          Expanded(
            flex: 1,
            child: Align(
              alignment: Alignment.centerRight,
              child: Column(
                crossAxisAlignment:
CrossAxisAlignment.end,
                children: [
                  Text(
                    "\P ${pizza.discount}",
                    style: TextStyle(
                      fontSize: 20,
                      fontWeight:
FontWeight.bold,
                      color: Theme.of(context)
                        .colorScheme
                        .primary),
                  ),
                  Text(
                    "\P ${pizza.price}",
                    style: const TextStyle(
                      fontSize: 16,
                      fontWeight:
FontWeight.bold,
                      color: Colors.grey,
                      decoration:
TextDecoration.lineThrough),
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    ],
  ),
),

```

```

const SizedBox(
  height: 12,
),
Row(
  children: [
    MyMacroWidget(
      title: "ККал",
      value: pizza.macros.calories,
      icon: FontAwesomeIcons.fire,
    ),
    const SizedBox(
      width: 10,
    ),
    MyMacroWidget(
      title: "Белка",
      value: pizza.macros.proteins,
      icon: FontAwesomeIcons.dumbbell,
    ),
    const SizedBox(
      width: 10,
    ),
    MyMacroWidget(
      title: "Жиров",
      value: pizza.macros.fat,
      icon: FontAwesomeIcons.oilWell,
    ),
    const SizedBox(
      width: 10,
    ),
    MyMacroWidget(
      title: "Угл",
      value: pizza.macros.carbs,
      icon: FontAwesomeIcons.breadSlice,
    ),
  ],
),
const SizedBox(
  height: 40,
),
SizedBox(
  width: MediaQuery.of(context).size.width,
  height: 50,
  child: TextButton(
    onPressed: () {},
    style: TextButton.styleFrom(
      elevation: 3.0,
      backgroundColor: Colors.black,
      foregroundColor: Colors.white,
      shape: RoundedRectangleBorder(
        borderRadius:
BorderRadius.circular(10))),
    child: const Text(

```


2 Безопасность жизнедеятельности

На всех этапах жизненного цикла проекта, будь то его разработка, внедрение или эксплуатация, важно обеспечить безопасность человека. Чтобы создать безопасные и комфортные условия работы, необходимо правильно оценить опасные и вредные факторы, связанные с производственной средой, физической и умственной нагрузкой, а также нервно-эмоциональным напряжением. Различные причины могут приводить к схожим изменениям в организме человека, поэтому важно принимать их во внимание и предпринимать соответствующие меры для обеспечения безопасности.

2.1 Потенциально опасные факторы при работе с ПЭВМ

Производственные факторы, которые представляют опасность и вредность, могут быть разделены на следующие группы по их природе: физические, химические, психофизиологические и биологические. При работе с компьютером (ПЭВМ) на программиста(пользователя) могут негативно влиять различные физические факторы, такие как изменение температуры воздуха (повышенная или пониженная), наличие избыточной пыли или газов в воздухе, избыточная или недостаточная влажность, высокий уровень шума, наличие ионизирующего излучения, риск поражения электрическим током, статическое электричество, плохое качество экрана дисплея и недостаточное освещение рабочего места. К химическим опасным факторам, которые могут постоянно воздействовать на программиста, относятся активные частицы, возникающие при ионизации воздуха в процессе работы на компьютере. В данном помещении отсутствуют биологические вредные производственные факторы. К психологическим вредным факторам, которые могут оказывать воздействие на программиста в течение рабочей смены, можно отнести нервно-эмоциональные перегрузки, умственное напряжение и перенапряжение зрительного анализатора.

					ПГУ1.090301.08.001 ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

2.2 Классификация помещения

Кабинет программиста, относящийся к категории В1-В4 (пожароопасность) I категории (выполняются легкие физические работы), должен соответствовать следующим требованиям: температура воздуха должна быть 22 °С (допустимый диапазон - 20-24 °С), оптимальная относительная влажность составляет 40-60% (допустимый предел - не более 75%), а скорость движения воздуха не должна превышать 0.1 м/с. Для обеспечения и автоматического поддержания оптимальных значений температуры, влажности, чистоты и скорости движения воздуха на рабочем месте, независимо от внешних условий, используется система водяного отопления в холодное время года и кондиционирование воздуха в теплое время года. Кондиционер, как часть вентиляционной системы, автоматически регулирует параметры воздушной среды в помещении. Владелец предприятия должен обеспечить следующие меры: поддержание работоспособности электрооборудования и его эксплуатацию в соответствии с требованиями нормативной технической документации, своевременное и качественное проведение профилактических работ, ремонта, модернизации и реконструкции энергетического оборудования, надежность работы электроустановок и безопасность их обслуживания, предотвращение использования технологий и методов работы, которые могут негативно влиять на окружающую среду, учет и анализ нарушений в работе электроустановок, несчастных случаев и принятие мер для устранения их причин, разработку должностных инструкций и производственных инструкций для электротехнического персонала, а также выполнение предписаний органов Госэнергонадзора.

2.3 Санитарно-гигиеническая характеристика

Ключевыми показателями, которые характеризуют микроклимат, являются следующие: температура воздуха, скорость движения воздуха и относительная влажность. Работа программиста категории IA предполагает соблюдение оптимальных требований и норм, включающих следующие значения:

					ПГУ1.090301.08.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

оптимальная температура воздуха составляет 22-24 °С, относительная влажность - 40-60%, а скорость движения воздуха должна быть 0,1 м/с. При работе с компьютером программист проводит продолжительное время в статическом положении и часто напрягает глаза. Так как работа за компьютером обычно происходит при искусственном освещении, необходимо обеспечить комфортные условия для сотрудников, чтобы уменьшить нагрузку на глаза. Отсутствие или недостаток освещения в помещении может негативно сказаться на здоровье глаз, снизить работоспособность, ухудшить кровообращение, вызвать сильное напряжение и доставить человеку дискомфорт. Это также может стать причиной сильных головных болей. После долгого рабочего дня возникают чувство усталости и слезоточивость глаз. При работе за компьютером следует соблюдать следующие требования: использовать лампы для освещения рабочего места; сочетать локальное и общее освещение; не позволять прямому и яркому попаданию солнечных лучей на монитор, так как это может повлиять на отображаемую картинку и вызвать искажения; не работать за компьютером в темных помещениях; помещения с компьютерами не должны находиться рядом с помещениями, где уровень шума и вибраций превышает нормальные значения; уровень шума на рабочем месте не должен превышать 50 дБА.

2.4 Расчет освещенности рабочего места

Правильный выбор и расчет освещенности рабочего места обеспечивает создание нормальных условий для зрения обслуживающего персонала, способствует повышению производительности труда. Освещение, соответствующее санитарным нормам, является главным условием гигиены труда и культуры производства. При хорошем освещении устраняется напряжение зрения, ускоряется темп работы.

Для рационального освещения необходимо выполнение следующих условий:

– постоянная освещенность рабочих поверхностей;

					ПГУ1.090301.08.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

- достаточная и равномерно распределенная яркость освещаемых рабочих поверхностей;
- отсутствие резких контрастов между яркостью рабочей поверхностью и окружающего пространства.

Для рационального освещения производственных помещений и рабочих мест большое значение имеет выбор цвета для окраски потолка, стен и производственного оборудования. Рассчитаем люминесцентное освещение кабинета, предназначенного для выполнения работ с размером объекта различения от 0,3 мин до 1 мин. Размеры помещений: A (длина)=4 м, B (ширина)=3 м, H (высота)=3 м. Площадь S помещения определяем по формуле 1.

$$S=A*B. \quad (1)$$

Подставив значения, получим:

$$S=4*3=12 \text{ м}^2.$$

Освещение проектируется при помощи светильников ШОД минимальной освещенностью $E_{\text{мин}}=200$ лк, средней удельной мощностью 17- 23 Вт/м². Высота подвеса над рабочей поверхностью $H_p=2$ м.

Освещение выполняется лампами ЛБ-2*40, световой поток ламп $F_{\text{л}}=2480$ лм, длина 1.2 м, коэффициент запаса равен $k=1.5$. Определим показатель помещения F по формуле 2.

$$F=A*B/H_p(A+B). \quad (2)$$

Подставив значения, получим:

$$F=4*3/2*(4+3) = 0.86.$$

Затем, для $F=0,86$, коэффициентов отражения потолка $P_{\text{п}}=0,7$, стен $P_{\text{с}}=0,5$ и расчетной плоскости $P_{\text{р}}=0,3$ находим коэффициент использования светового потока $h=0,37$ по формуле 3.

$$N=E_{\text{мин}}*S*K/F_{\text{л}} *h*Z*n, \quad (3)$$

где $n=2$ общее число ламп в светильнике.

Подставив значения, получим:

$$N=200*12*1,5/2480*0,37*0.9*2=2 \text{ штуки.}$$

Общее количество ламп равно $n=2*2=4$.

Таким образом, выполнение перечисленных выше требований к организации рабочего места обеспечит комфортные условия умственного труда.

					ПГУ1.090301.08.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В результате проделанной работы было успешно разработано кроссплатформенное мобильное приложение для кафе, которое обладает удобным и интуитивно понятным пользовательским интерфейсом. Данный проект представляет собой мобильное приложение для iOS и Android, который включает все необходимые функции и соответствует требованиям поставленной задаче. Разработка данного приложения позволило значительно улучшить навыки разработки мобильных приложений, освоить среду разработки Visual Studio Code, а также расширить знания о работе с базой данных Firebase.

В дальнейшем можно усовершенствовать разработанное мобильное приложение, добавить возможность публиковать промо-материалы, возможность фильтрации продуктов и разные экраны для разных товаров.

					ПГУ1.090301.08.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. [Электронный ресурс] / режим доступа: <https://metanit.com/dart/flutter/>, свободный. (дата обращения 10.05.2024).
2. Заметти Франк – Flutter на практике (дата обращения 26.04.2024).
3. Андрей Алеев – Быстрый старт Flutter-разработчика (дата обращения 27.04.2024).
4. Барри Берд – Флаттер для чайников (дата обращения 28.04.2024).
5. Санитарно-гигиенические рекомендации для организации рабочего места пользователя персональным компьютером. [Электронный ресурс] / Режим доступа: <https://sakhgig.ru/main/info/664-sanitarno-gigienicheskie-rekomendacii-dlyaorganizacii-rabochego-mesta-polzovatelya-personalnym-kompyuterom.html>, свободный. (дата обращения 25.05.2024).

Приложение А

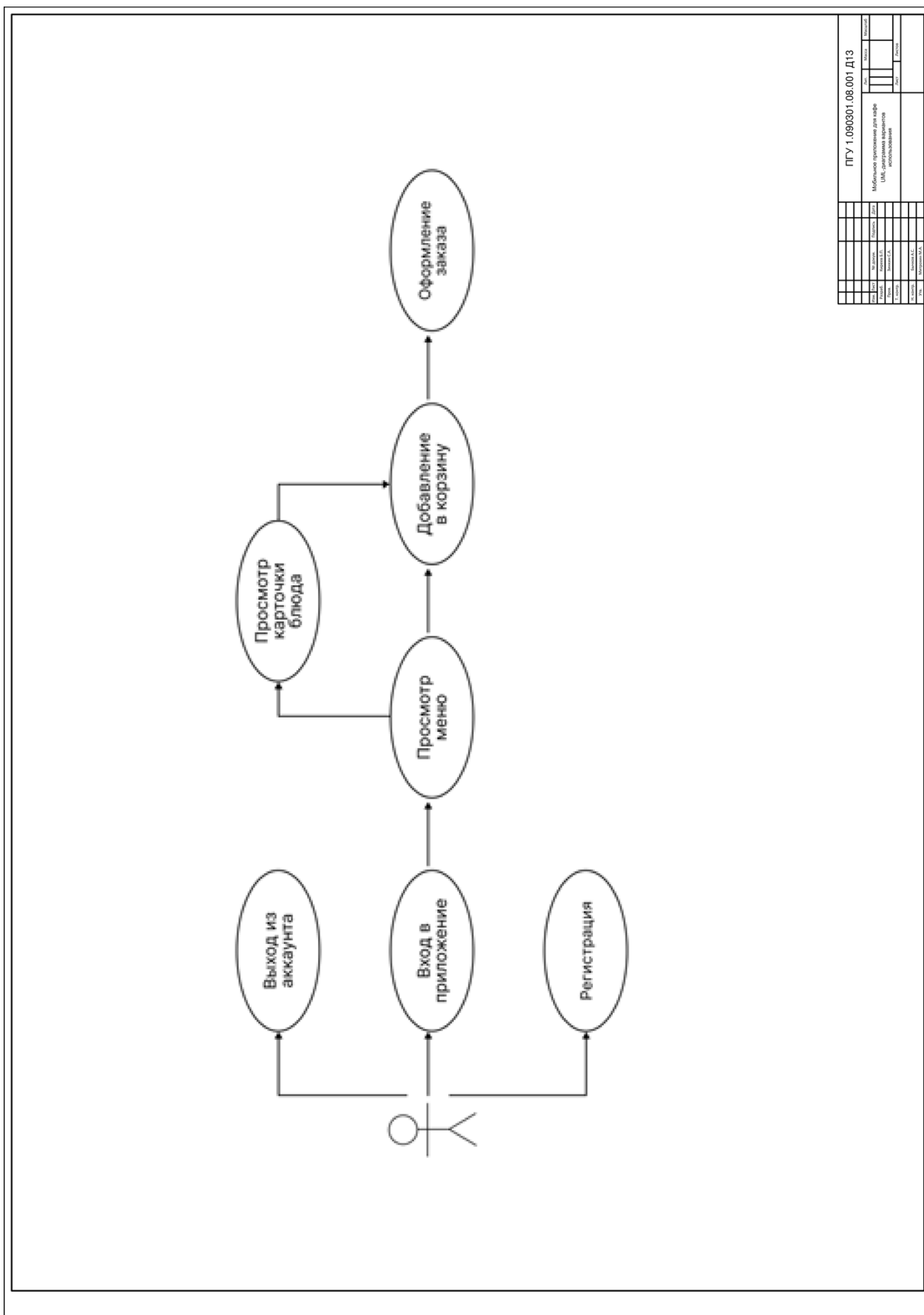
(Обязательное)

Мобильное приложение для кафе

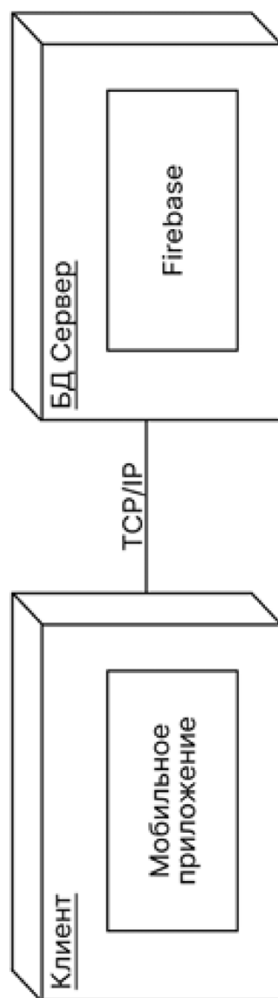
UML-диаграммы

					ПГУ1.090301.08.001 ПЗ	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – UML-диаграмма вариантов использования



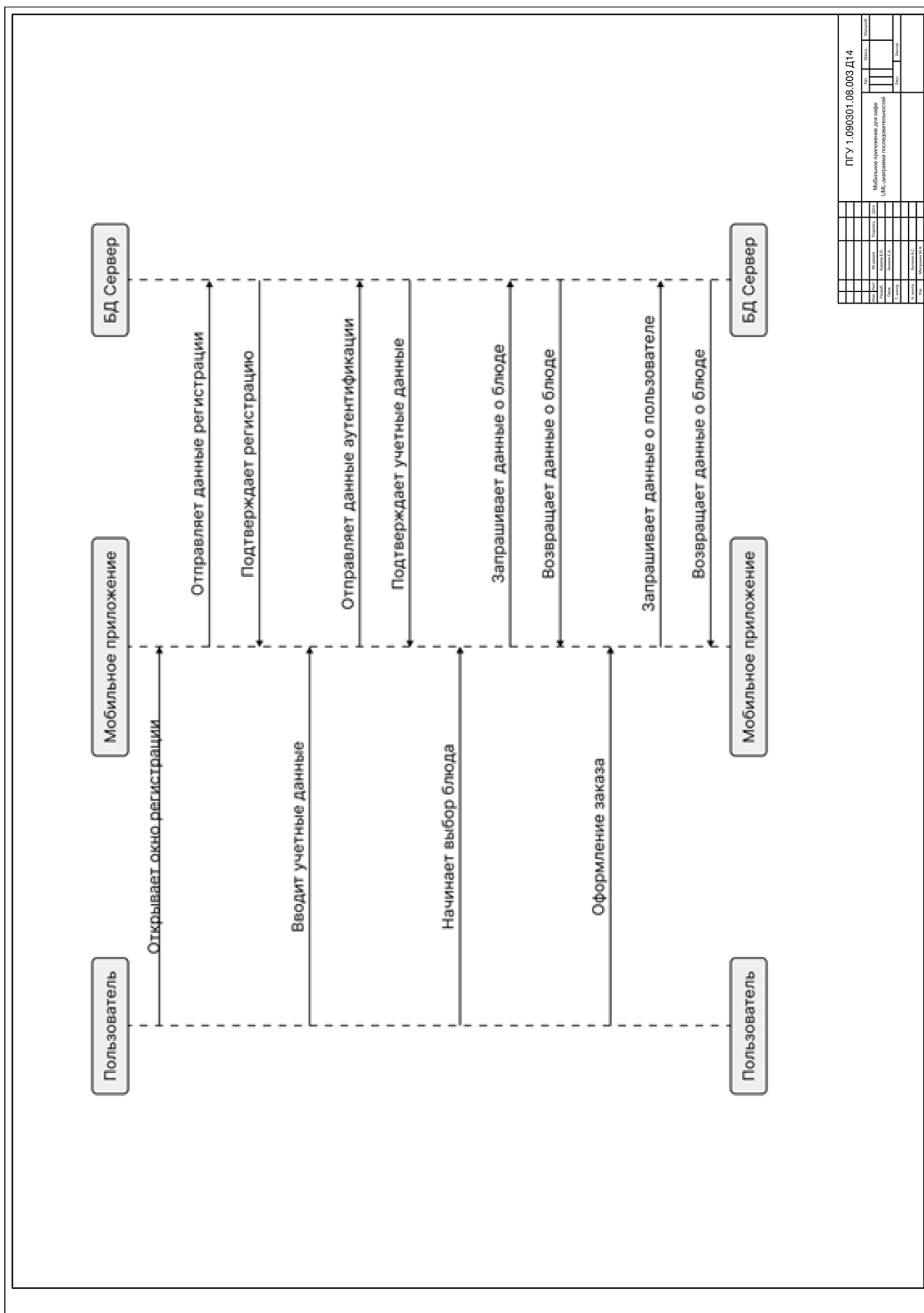
Приложение А.2 – UML-диаграмма развёртывания

[illegible]

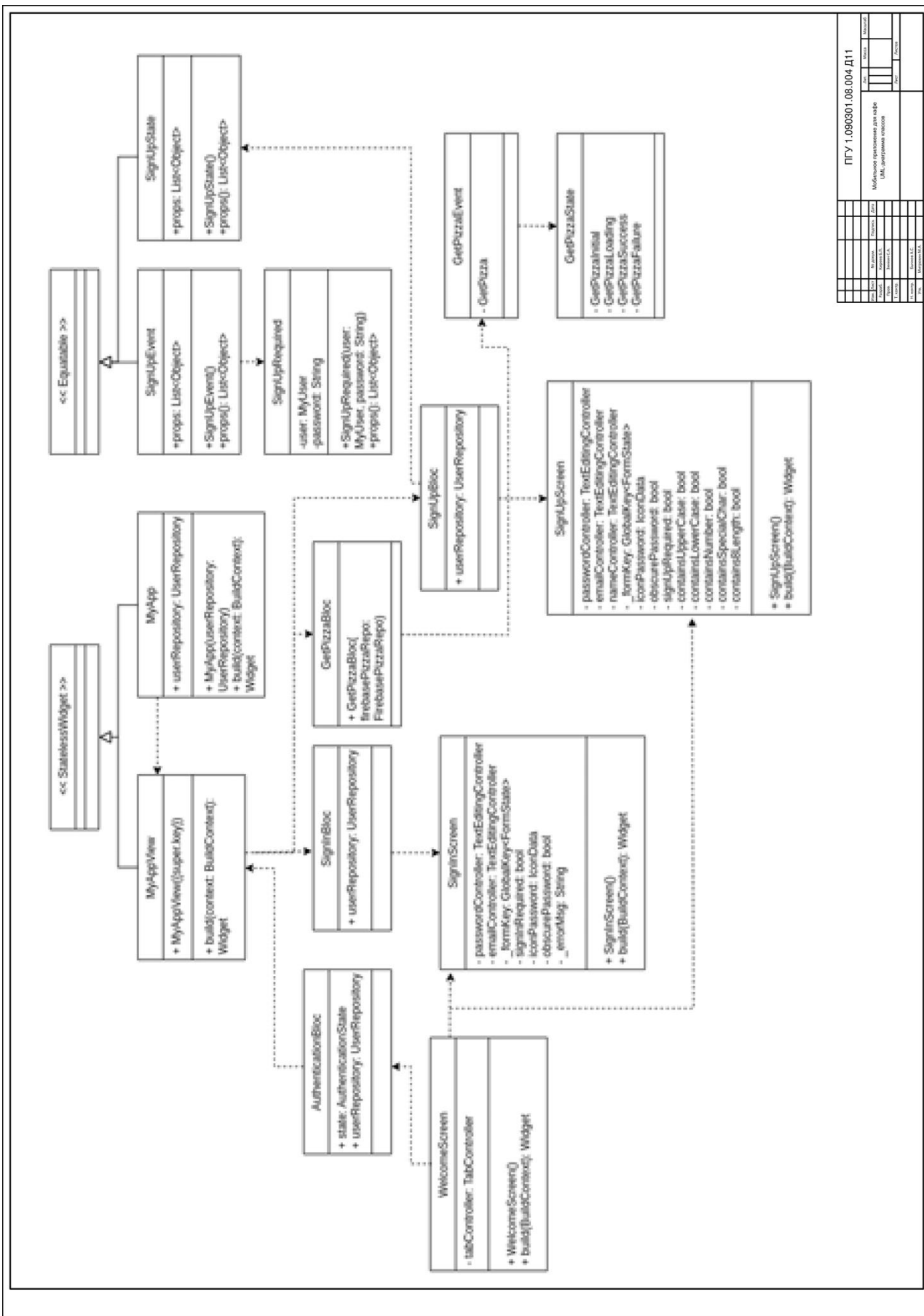
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.08.001 ПЗ

Приложение А.3 – UML-диаграмма последовательностей

[illegible]

Приложение А.4 – UML-диаграмма классов



Приложение Б

(Обязательное)

Мобильное приложение для кафе

Листинг программы

					<i>ПГУ1.090301.08.001 ПЗ</i>	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

Файл main.dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:pizza_app/app.dart';
import 'package:pizza_app/simple_bloc_observer.dart';
import 'package:user_repository/user_repository.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  Bloc.observer = SimpleBlocObserver();
  runApp(MyApp(FirebaseUserRepo()));
}
```

Файл app.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:user_repository/user_repository.dart';
import 'app_view.dart';
import 'blocs/authentication_bloc/authentication_bloc.dart';

class MyApp extends StatelessWidget {
  final UserRepository userRepository;
  const MyApp(this.userRepository, {super.key});

  @override
  Widget build(BuildContext context) {
    return RepositoryProvider<AuthenticationBloc>(
      create: (context) => AuthenticationBloc(userRepository:
userRepository),
      child: const MyAppView(),
    );
  }
}
```

Файл app_view.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import
'package:pizza_app/blocs/authentication_bloc/authentication_bloc
.dart';
import
'package:pizza_app/screens/auth/blocs/sign_in_bloc/sign_in_bloc.
dart';
import
'package:pizza_app/screens/home/blocs/get_pizza_bloc/get_pizza_b
loc.dart';
import 'package:pizza_repository/pizza_repository.dart';
```

					ПГУ1.090301.08.001 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

```

import 'screens/auth/views/welcome_screen.dart';
import 'screens/home/views/home_screen.dart';

class MyAppView extends StatelessWidget {
  const MyAppView({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Pizza Delivery',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.light(
          background: Colors.grey.shade200,
          onBackground: Colors.black,
          primary: Colors.blue,
          onPrimary: Colors.white)),
      home: BlocBuilder<AuthenticationBloc,
AuthenticationState>(
        builder: ((context, state) {
          if (state.status ==
AuthenticationStatus.authenticated) {
            return MultiBlocProvider(
              providers: [
                BlocProvider(
                  create: (context) => SignInBloc(
context.read<AuthenticationBloc>().userRepository),
                ),
                BlocProvider(
                  create: (context) =>
GetPizzaBloc(FirebasePizzaRepo())..add(GetPizza()),
                ),
              ],
              child: const HomeScreen(),
            );
          } else {
            return const WelcomeScreen();
          }
        })),
    );
  }
}

```

Файл app_view.dart

```

import 'dart:developer';
import 'package:bloc/bloc.dart';

class SimpleBlocObserver extends BlocObserver {

```

					ПГУ1.090301.08.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

```

@override
void onCreate(BlocBase bloc) {
  super.onCreate(bloc);
  log('onCreate -- bloc: ${bloc.runtimeType}');
}

@override
void onEvent(Bloc bloc, Object? event) {
  super.onEvent(bloc, event);
  log('onEvent -- bloc: ${bloc.runtimeType}, event: $event');
}

@override
void onChange(BlocBase bloc, Change change) {
  super.onChange(bloc, change);
  log('onChange -- bloc: ${bloc.runtimeType}, change: $change');
}

@override
void onTransition(Bloc bloc, Transition transition) {
  super.onTransition(bloc, transition);
  log('onTransition -- bloc: ${bloc.runtimeType}, transition: $transition');
}

@override
void onError(BlocBase bloc, Object error, StackTrace stackTrace) {
  log('onError -- bloc: ${bloc.runtimeType}, error: $error');
  super.onError(bloc, error, stackTrace);
}

@override
void onClose(BlocBase bloc) {
  super.onClose(bloc);
  log('onClose -- bloc: ${bloc.runtimeType}');
}
}

```

Файл welcome_screen.dart

```

import 'dart:ui';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import
'../../blocs/authentication_bloc/authentication_bloc.dart';
import '../blocs/sign_up_bloc/sign_up_bloc.dart';
import '../blocs/sign_in_bloc/sign_in_bloc.dart';
import 'sign_in_screen.dart';
import 'sign_up_screen.dart';

```

```

class WelcomeScreen extends StatefulWidget {
  const WelcomeScreen({super.key});

  @override
  State<WelcomeScreen> createState() => _WelcomeScreenState();
}

class _WelcomeScreenState extends State<WelcomeScreen>
  with TickerProviderStateMixin {
  late TabController tabController;

  @override
  void initState() {
    tabController = TabController(initialIndex: 0, length: 2,
vsync: this);
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Theme.of(context).colorScheme.background,
      body: SingleChildScrollView(
        child: SizedBox(
          height: MediaQuery.of(context).size.height,
          child: Stack(
            children: [
              Align(
                alignment: const AlignmentDirectional(20, -1.2),
                child: Container(
                  height: MediaQuery.of(context).size.width,
                  width: MediaQuery.of(context).size.width,
                  decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    color:
Theme.of(context).colorScheme.tertiary),
                ),
              ),
              Align(
                alignment: const AlignmentDirectional(2.7, -
1.2),
                child: Container(
                  height: MediaQuery.of(context).size.width /
1.3,
                  width: MediaQuery.of(context).size.width /
1.3,
                  decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    color:
Theme.of(context).colorScheme.primary),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}

```

```

        BackdropFilter(
          filter: ImageFilter.blur(sigmaX: 100.0, sigmaY:
100.0),
          child: Container(),
        ),
        Align(
          alignment: Alignment.center,
          child: SizedBox(
            height: MediaQuery.of(context).size.height /
1.8,
            child: Column(
              children: [
                Padding(
                  padding: const
EdgeInsets.symmetric(horizontal: 50.0),
                  child: TabBar(
                    controller: tabController,
                    unselectedLabelColor:
Theme.of(context)
                      .colorScheme
                      .onBackground
                      .withOpacity(0.5),
                    labelColor:
Theme.of(context).colorScheme.onBackground,
                    tabs: const [
                      Padding(
                        padding: EdgeInsets.all(12.0),
                        child: Text(
                          'Вход',
                          style: TextStyle(
                            fontSize: 18,
                          ),
                        ),
                      ),
                      Padding(
                        padding: EdgeInsets.all(8.0),
                        child: Text(
                          'Регистрация',
                          style: TextStyle(
                            fontSize: 18,
                          ),
                        ),
                      ),
                    ],
                  ),
                Expanded(
                  child: TabBarView(
                    controller: tabController,
                    children: [
                      BlocProvider<SignInBloc>(

```

Файл sign_in_screen.dart

ПГУ1.090301.08.001 ПЗ

```

@override
Widget build(BuildContext context) {
  return BlocListener<SignInBloc, SignInState>(
    listener: (context, state) {
      if (state is SignInSuccess) {
        setState(() {
          signInRequired = false;
        });
      } else if (state is SignInProcess) {
        setState(() {
          signInRequired = true;
        });
      } else if (state is SignInFailure) {
        setState(() {
          signInRequired = false;
          _errorMsg = 'Неправильная почта или пароль';
        });
      }
    },
    child: Form(
      key: _formKey,
      child: Column(
        children: [
          const SizedBox(height: 20),
          SizedBox(
            width: MediaQuery.of(context).size.width *
0.9,
            child: MyTextField(
              controller: emailController,
              hintText: 'Почта',
              obscureText: false,
              keyboardType: TextInputType.emailAddress,
              prefixIcon: const
Icon(CupertinoIcons.mail_solid),
              errorMsg: _errorMsg,
              validator: (val) {
                if (val!.isEmpty) {
                  return 'Пожалуйста заполните поле';
                } else if (!RegExp(r'^[\w-\.] +@([\w-
]+.)+[\w-]{2,4}$').
                .hasMatch(val)) {
                  return 'Пожалуйста введите правильную
почту';
                }
                return null;
              }
            )),
          const SizedBox(height: 10),
          SizedBox(
            width: MediaQuery.of(context).size.width * 0.9,
            child: MyTextField(
              controller: passwordController,
              hintText: 'Пароль',

```

```

        obscureText: obscurePassword,
        keyboardType: TextInputType.visiblePassword,
        prefixIcon: const
Icon(CupertinoIcons.lock_fill),
        errorMsg: _errorMsg,
        validator: (val) {
            if (val!.isEmpty) {
                return 'Пожалуйста заполните поле';
            } else if (!RegExp(
                r'^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-
9])(?=.*?[!@#\$&*~`)\%\-(\_+=;:,.<>/?"[\]\}\|\^]).{8,}$')
                .hasMatch(val)) {
                return 'Пожалуйста введите пароль';
            }
            return null;
        },
        suffixIcon: IconButton(
            onPressed: () {
                setState(() {
                    obscurePassword = !obscurePassword;
                    if (obscurePassword) {
                        iconPassword =
CupertinoIcons.eye_fill;
                    } else {
                        iconPassword =
CupertinoIcons.eye_slash_fill;
                    }
                });
            },
            icon: Icon(iconPassword),
        ),
    ),
),
const SizedBox(height: 20),
!signInRequired
? SizedBox(
    width: MediaQuery.of(context).size.width *
0.5,
    child: TextButton(
        onPressed: () {
            if
(_formKey.currentState!.validate()) {
context.read<SignInBloc>().add(SignInRequired(
                    emailController.text,
                    passwordController.text));
            }
        },
        style: TextButton.styleFrom(
            elevation: 3.0,
            backgroundColor:

```

```

Theme.of(context).colorScheme.primary,
        foregroundColor: Colors.white,
        shape: RoundedRectangleBorder(
            borderRadius:
BorderRadius.circular(60))),
        child: const Padding(
            padding: EdgeInsets.symmetric(
                horizontal: 25, vertical: 5),
            child: Text(
                'Вход',
                textAlign: TextAlign.center,
                style: TextStyle(
                    color: Colors.white,
                    fontSize: 16,
                    fontWeight: FontWeight.w600),
            ),
        )),
    ),
    : const CircularProgressIndicator(),
    ],
    )),
    );
}
}

```

Файл sign_up_screen.dart

```

import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:user_repository/user_repository.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import '../components/my_text_field.dart';
import '../blocs/sign_up_bloc/sign_up_bloc.dart';

class SignUpScreen extends StatefulWidget {
  const SignUpScreen({super.key});

  @override
  State<SignUpScreen> createState() => _SignUpScreenState();
}

class _SignUpScreenState extends State<SignUpScreen> {
  final passwordController = TextEditingController();
  final emailController = TextEditingController();
  final nameController = TextEditingController();
  final _formKey = GlobalKey<FormState>();
  IconData iconPassword = CupertinoIcons.eye_fill;
  bool obscurePassword = true;
  bool signUpRequired = false;

  bool containsUpperCase = false;

```

```

bool containsLowerCase = false;
bool containsNumber = false;
bool containsSpecialChar = false;
bool contains8Length = false;

@override
Widget build(BuildContext context) {
  return BlocListener<SignUpBloc, SignUpState>(
    listener: (context, state) {
      if (state is SignUpSuccess) {
        setState(() {
          signUpRequired = false;
        });
      } else if (state is SignUpProcess) {
        setState(() {
          signUpRequired = true;
        });
      } else if (state is SignUpFailure) {
        return;
      }
    },
    child: Form(
      key: _formKey,
      child: Center(
        child: Column(
          children: [
            const SizedBox(height: 20),
            SizedBox(
              width: MediaQuery.of(context).size.width * 0.9,
              child: MyTextField(
                controller: emailController,
                hintText: 'Почта',
                obscureText: false,
                keyboardType: TextInputType.emailAddress,
                prefixIcon: const
Icon(CupertinoIcons.mail_solid),
                validator: (val) {
                  if (val!.isEmpty) {
                    return 'Пожалуйста введите поле';
                  } else if (!RegExp(r'^[\w-\.]++@([\w-
]+.)([\w-]{2,4})$').hasMatch(val)) {
                    return 'Пожалуйста введите правильный
email';
                  }
                  return null;
                })),
            const SizedBox(height: 10),
            SizedBox(
              width: MediaQuery.of(context).size.width * 0.9,
              child: MyTextField(

```

```
Icon(CupertinoIcons.lock_fill),
```

ПГУ1.090301.08.001 ПЗ

```

        });
    }
    return null;
},
suffixIcon: IconButton(
    onPressed: () {
        setState(() {
            obscurePassword = !obscurePassword;
            if (obscurePassword) {
                iconPassword =
CupertinoIcons.eye_fill;
            } else {
                iconPassword =
CupertinoIcons.eye_slash_fill;
            }
        });
    },
    icon: Icon(iconPassword),
),
validator: (val) {
    if (val!.isEmpty) {
        return 'Пожалуйста введите поле';
    } else if (!RegExp(
        r'^(?=.*?[A-Z])(?=.*?[a-
z])(?=.*?[0-9])(?=.*?[!@#\$&*~`)\%\-
(_+=;:,.<>/?"[\]\|\^]).{8,}$')
        .hasMatch(val)) {
        return 'Пожалуйста введите правильный
пароль';
    }
    return null;
}),
),
const SizedBox(height: 10),
Row(
    mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
        Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
                Text(
                    "● 1 uppercase",
                    style: TextStyle(
                        color: containsUpperCase
                            ? Colors.green
                            :
Theme.of(context).colorScheme.onBackground),
                ),
                Text(

```

```

        "● 1 lowercase",
        style: TextStyle(
          color: containsLowerCase
            ? Colors.green
            :
Theme.of(context).colorScheme.onBackground),
      ),
      Text(
        "● 1 number",
        style: TextStyle(
          color: containsNumber
            ? Colors.green
            :
Theme.of(context).colorScheme.onBackground),
      ),
    ],
  ),
  Column(
    crossAxisAlignment:
CrossAxisAlignment.start,
    children: [
      Text(
        "● 1 special character",
        style: TextStyle(
          color: containsSpecialChar
            ? Colors.green
            :
Theme.of(context).colorScheme.onBackground),
      ),
      Text(
        "● 8 minimum character",
        style: TextStyle(
          color: contains8Length
            ? Colors.green
            :
Theme.of(context).colorScheme.onBackground),
      ),
    ],
  ),
  ),
  ],
),
const SizedBox(height: 10),
SizedBox(
  width: MediaQuery.of(context).size.width * 0.9,
  child: MyTextField(
    controller: nameController,
    hintText: 'Имя',
    obscureText: false,
    keyboardType: TextInputType.name,
    prefixIcon: const
Icon(CupertinoIcons.person_fill),

```

```

        validator: (val) {
          if (val!.isEmpty) {
            return 'Пожалуйста заполните поле';
          } else if (val.length > 30) {
            return 'Имя слишком длинное';
          }
          return null;
        })),
      ),
      SizedBox(height:
MediaQuery.of(context).size.height * 0.02),
      !signUpRequired
        ? SizedBox(
          width: MediaQuery.of(context).size.width *
0.5,
          child: TextButton(
            onPressed: () {
              if
(_formKey.currentState!.validate()) {
                MyUser myUser = MyUser.empty;
                myUser.email =
emailController.text;
                myUser.name = nameController.text;
                setState(() {
context.read<SignUpBloc>().add(SignUpRequired(
myUser,
passwordController.text));
                });
              }
            },
            style: TextButton.styleFrom(
              elevation: 3.0,
              backgroundColor:
Theme.of(context).colorScheme.primary,
              foregroundColor: Colors.white,
              shape: RoundedRectangleBorder(
                borderRadius:
BorderRadius.circular(60))),
            child: const Padding(
              padding: EdgeInsets.symmetric(
                horizontal: 25, vertical: 5),
              child: Text(
                'Регистрация',
                textAlign: TextAlign.center,
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 16,
                  fontWeight: FontWeight.w600),
              ),
            ),

```

```

       )),
    ),
    : const CircularProgressIndicator(),
  ],
),
),
),
);
}
}

```

Файл sign_in_bloc.dart

```

import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';
import 'package:user_repository/user_repository.dart';

part 'sign_in_event.dart';
part 'sign_in_state.dart';

class SignInBloc extends Bloc<SignInEvent, SignInState> {
  final UserRepository _userRepository;

  SignInBloc(this._userRepository) : super(SignInInitial()) {
    on<SignInRequired>((event, emit) async {
      emit(SignInProcess());
      try {
        await _userRepository.signIn(event.email,
event.password);
      } catch (e) {
        emit(SignInFailure());
      }
    });

    on<SignOutRequired>((event, emit) async => await
_userRepository.logOut());
  }
}

```

Файл sign_in_event.dart

```

part of 'sign_in_bloc.dart';

sealed class SignInEvent extends Equatable {
  const SignInEvent();

  @override
  List<Object> get props => [];
}

class SignInRequired extends SignInEvent {
  final String email;
  final String password;
}

```

```

    const SignInRequired(this.email, this.password);

    @override
    List<Object> get props => [email, password];
}

```

```

class SignOutRequired extends SignInEvent {}

```

Файл **sign_in_state.dart**

```

part of 'sign_in_bloc.dart';

sealed class SignInState extends Equatable {
  const SignInState();

  @override
  List<Object> get props => [];
}

final class SignInInitial extends SignInState {}

final class SignInFailure extends SignInState {}

final class SignInProcess extends SignInState {}

final class SignInSuccess extends SignInState {}

```

Файл **sign_up_bloc.dart**

```

import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';
import 'package:user_repository/user_repository.dart';

part 'sign_up_event.dart';
part 'sign_up_state.dart';

class SignUpBloc extends Bloc<SignUpEvent, SignUpState> {
  final UserRepository _userRepository;

  SignUpBloc(this._userRepository) : super(SignUpInitial()) {
    on<SignUpRequired>((event, emit) async {
      emit(SignUpProcess());
      try {
        MyUser myUser =
          await _userRepository.signUp(event.user,
event.password);
        await _userRepository.setUserData(myUser);
        emit(SignUpSuccess());
      } catch (e) {
        emit(SignUpFailure());
      }
    });
  }
}

```

```

    }
  });
}
}

```

Файл sign_up_event.dart

```

part of 'sign_up_bloc.dart';

sealed class SignUpEvent extends Equatable {
  const SignUpEvent();

  @override
  List<Object> get props => [];
}

class SignUpRequired extends SignUpEvent {
  final MyUser user;
  final String password;

  const SignUpRequired(this.user, this.password);

  @override
  List<Object> get props => [user, password];
}

```

Файл sign_up_state.dart

```

part of 'sign_up_bloc.dart';

sealed class SignUpState extends Equatable {
  const SignUpState();

  @override
  List<Object> get props => [];
}

final class SignUpInitial extends SignUpState {}

class SignUpSuccess extends SignUpState {}

class SignUpFailure extends SignUpState {}

class SignUpProcess extends SignUpState {}

```

Файл home_screen.dart

```

import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

```

```

import
'package:pizza_app/screens/auth/blocs/sign_in_bloc/sign_in_bloc.
dart';
import
'package:pizza_app/screens/home/blocs/get_pizza_bloc/get_pizza_b
loc.dart';
import
'package:pizza_app/screens/home/views/details_screen.dart';

class HomeScreen extends StatelessWidget {
  const HomeScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Theme.of(context).colorScheme.background,
      appBar: AppBar(
        backgroundColor:
Theme.of(context).colorScheme.background,
        title: Row(
          children: [
            Image.asset('assets/8.png', scale: 14),
            const SizedBox(width: 8),
            const Text(
              'Неаполи',
              style: TextStyle(fontWeight: FontWeight.w900,
fontSize: 30),
            )
          ],
        ),
        actions: [
          IconButton(onPressed: () {}, icon: const
Icon(CupertinoIcons.cart)),
          IconButton(
            onPressed: () {

context.read<SignInBloc>().add(SignOutRequired());
            },
            icon: const
Icon(CupertinoIcons.arrow_right_to_line)),
        ],
        body: Padding(
          padding: const EdgeInsets.all(16.0),
          child: BlocBuilder<GetPizzaBloc, GetPizzaState>(
            builder: (context, state) {
              if (state is GetPizzaSuccess) {
                return GridView.builder(
                  gridDelegate: const
SliverGridDelegateWithFixedCrossAxisCount(
                    crossAxisCount: 2,
                    crossAxisSpacing: 16,

```

```

        mainAxisAlignmentSpacing: 16,
        childAspectRatio: 9 / 16),
    itemCount: state.pizzas.length,
    itemBuilder: (context, int i) {
      return Material(
        elevation: 3,
        color: Colors.white,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(20),
        ),
        child: InkWell(
          borderRadius: BorderRadius.circular(20),
          onTap: () {
            Navigator.push(
              context,
              MaterialPageRoute<void>(
                builder: (BuildContext context) =>
DetailsScreen(state.pizzas[i]),
              ),
            );
          },
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
Image.network(state.pizzas[i].picture),
          Padding(
            padding:
              const
EdgeInsets.symmetric(horizontal: 12.0),
            child: Row(
              children: [
                Container(
                  decoration: BoxDecoration(
                    color:
state.pizzas[i].isVeg
                      ? Colors.green
                      : Colors.red,
                    borderRadius:
BorderRadius.circular(30)),
                  child: Padding(
                    padding: const
EdgeInsets.symmetric(
                      vertical: 4,
                      horizontal: 8),
                    child: Text(
                      state.pizzas[i].isVeg
                        ? "Веганская"
                        : "Мясная",

```

```

FontWeight.bold,
Colors.green.withOpacity(0.2),
BorderRadius.circular(30)),
EdgeInsets.symmetric(
horizontal: 8),
1
state.pizzas[i].spicy == 2
state.pizzas[i].spicy == 1
state.pizzas[i].spicy == 2
Colors.orange
Colors.redAccent,
FontWeight.w800,
style: const TextStyle(
color: Colors.white,
fontWeight:
fontSize: 10),
),
),
),
const SizedBox(width: 8),
Container(
decoration: BoxDecoration(
color:
borderRadius:
child: Padding(
padding: const
vertical: 4,
child: Text(
state.pizzas[i].spicy ==
? "🌶️ Прямая"
:
? "🌶️ Баланс"
: "🌶️ Острая",
style: TextStyle(
color:
? Colors.green
:
?
:
fontWeight:
fontSize: 10),
),
),
),
const SizedBox(height: 8),
Padding(
padding:

```

```

                                const
EdgeInsets.symmetric(horizontal: 12.0),
                                child: Text(
                                    state.pizzas[i].name,
                                    style: const TextStyle(
                                        fontSize: 20, fontWeight:
EdgeInsets.symmetric(horizontal: 12.0),
                                child: Text(
                                    state.pizzas[i].description,
                                    style: TextStyle(
                                        fontSize: 10,
                                        color: Colors.grey.shade500,
                                    ),
                                ),
                                Padding(
                                    padding: const
EdgeInsets.symmetric(
                                    horizontal: 12.0),
                                child: Row(
                                    mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                                    children: [
                                        Row(
                                            children: [
                                                Text(
                                                    "\P
${state.pizzas[i].discount}",
                                                    style: TextStyle(
                                                        fontSize: 16,
                                                        color:
Theme.of(context)
                                                        .colorScheme
                                                        .primary,
                                                        fontWeight:
FontWeight.w700),
                                                ),
                                                const SizedBox(
                                                    width: 5,
                                                ),
                                                Text(
                                                    "\P
${state.pizzas[i].price}",
                                                    style: TextStyle(
                                                        fontSize: 12,

```

```

        color:
Colors.grey.shade500,
        fontWeight:
FontWeight.w700,
        decoration:
TextDecoration.lineThrough),
    ),
  ],
),
IconButton(
  onPressed: () {},
  icon: const Icon(
CupertinoIcons.add_circled_solid))
  ],
))
  ],
),
),
);
});
} else if (state is GetPizzaLoading) {
  return const Center(
    child: CircularProgressIndicator(),
  );
} else {
  return const Center(
    child: Text("An error has occurred..."),
  );
}
},
),
),
);
}
}

```

Файл details_screen.dart

```

import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:pizza_repository/pizza_repository.dart';
import '../components/macro.dart';

class DetailsScreen extends StatelessWidget {
  final Pizza pizza;
  const DetailsScreen(this.pizza, {super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(

```

```

        backgroundColor: Theme.of(context).colorScheme.background,
        appBar: AppBar(
          backgroundColor:
Theme.of(context).colorScheme.background,
        ),
        body: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 20),
          child: Column(
            children: [
              Container(
                width: MediaQuery.of(context).size.width,
                height: MediaQuery.of(context).size.width - (40),
                decoration: BoxDecoration(
                  color: Colors.white,
                  borderRadius: BorderRadius.circular(30),
                  boxShadow: const [
                    BoxShadow(
                      color: Colors.grey, offset: Offset(3,
3), blurRadius: 5)
                  ],
                image: DecorationImage(image:
NetworkImage(pizza.picture))),
              ),
              const SizedBox(
                height: 30,
              ),
              Container(
                decoration: BoxDecoration(
                  color: Colors.white,
                  borderRadius: BorderRadius.circular(30),
                  boxShadow: const [
                    BoxShadow(
                      color: Colors.grey, offset: Offset(3, 3),
blurRadius: 5)
                  ],
              ),
              child: Padding(
                padding: const EdgeInsets.all(20.0),
                child: Column(
                  children: [
                    Row(
                      mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                      children: [
                        Expanded(
                          flex: 2,
                          child: Text(
                            pizza.name,
                            style: const TextStyle(
                              fontSize: 20, fontWeight:
FontWeight.bold),
                          ),

```

```

),
Expanded(
  flex: 1,
  child: Align(
    alignment: Alignment.centerRight,
    child: Column(
      crossAxisAlignment:
CrossAxisAlignment.end,
      children: [
        Text(
          "\P ${pizza.discount}",
          style: TextStyle(
            fontSize: 20,
            fontWeight:
FontWeight.bold,
            color: Theme.of(context)
              .colorScheme
              .primary),
        ),
        Text(
          "\P ${pizza.price}",
          style: const TextStyle(
            fontSize: 16,
            fontWeight:
FontWeight.bold,
            color: Colors.grey,
            decoration:
TextDecoration.lineThrough),
        ),
      ],
    ),
  ),
),
const SizedBox(
  height: 12,
),
Row(
  children: [
    MyMacroWidget(
      title: "ККал",
      value: pizza.macros.calories,
      icon: FontAwesomeIcons.fire,
    ),
    const SizedBox(
      width: 10,
    ),
    MyMacroWidget(
      title: "Белка",
      value: pizza.macros.proteins,
      icon: FontAwesomeIcons.dumbbell,

```

```

    ),
    const SizedBox(
      width: 10,
    ),
    MyMacroWidget(
      title: "Жиров",
      value: pizza.macros.fat,
      icon: FontAwesomeIcons.oilWell,
    ),
    const SizedBox(
      width: 10,
    ),
    MyMacroWidget(
      title: "Угл",
      value: pizza.macros.carbs,
      icon: FontAwesomeIcons.breadSlice,
    ),
  ],
),
const SizedBox(
  height: 40,
),
SizedBox(
  width: MediaQuery.of(context).size.width,
  height: 50,
  child: TextButton(
    onPressed: () {},
    style: TextButton.styleFrom(
      elevation: 3.0,
      backgroundColor: Colors.black,
      foregroundColor: Colors.white,
      shape: RoundedRectangleBorder(
        borderRadius:
BorderRadius.circular(10))),
    child: const Text(
      "Добавить в корзину ",
      style: TextStyle(
        color: Colors.white,
        fontSize: 20,
        fontWeight: FontWeight.w600),
    ),
  ),
),
],
),
),
),
),
),
);
}

```

```
}
```

Файл get_pizza_bloc.dart

```
import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';
import 'package:pizza_repository/pizza_repository.dart';

part 'get_pizza_event.dart';
part 'get_pizza_state.dart';

class GetPizzaBloc extends Bloc<GetPizzaEvent, GetPizzaState> {
  final PizzaRepo _pizzaRepo;

  GetPizzaBloc(this._pizzaRepo) : super(GetPizzaInitial()) {
    on<GetPizza>((event, emit) async {
      emit(GetPizzaLoading());
      try {
        List<Pizza> pizzas = await _pizzaRepo.getPizzas();
        emit(GetPizzaSuccess(pizzas));
      } catch (e) {
        emit(GetPizzaFailure());
      }
    });
  }
}
```

Файл get_pizza_event.dart

```
part of 'get_pizza_bloc.dart';

sealed class GetPizzaEvent extends Equatable {
  const GetPizzaEvent();

  @override
  List<Object> get props => [];
}

class GetPizza extends GetPizzaEvent {}
```

Файл get_pizza_state.dart

```
part of 'get_pizza_bloc.dart';

sealed class GetPizzaState extends Equatable {
  const GetPizzaState();

  @override
  List<Object> get props => [];
}

final class GetPizzaInitial extends GetPizzaState {}
```

					ПГУ1.090301.08.001 ПЗ	Лист
						94
Изм.	Лист	№ докум.	Подпись	Дата		

```

final class GetPizzaFailure extends GetPizzaState {}

final class GetPizzaLoading extends GetPizzaState {}

final class GetPizzaSuccess extends GetPizzaState {
    final List<Pizza> pizzas;

    const GetPizzaSuccess(this.pizzas);

    @override
    List<Object> get props => [pizzas];
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(Обязательное)

Мобильное приложение для кафе

Презентация

					ПГУ1.090301.08.001 ПЗ	Лист
						96
Изм.	Лист	№ докум.	Подпись	Дата		

Разработка мобильного приложения для кафе

Автор работы:
студент группы 20ВВП1 Киреев Б.П.
Научный руководитель:
профессор кафедры ВТ Зинкин С.А.

Пенза, 2024

Актуальность проекта

**Цель — разработать
мобильное приложение для кафе**

Аналоги

Суши-бар «Дайкон» Кофейня «Кофейный экспресс» Шашлычная «Вертел»

					ПГУ1.090301.08.001 ПЗ	Лист
						98
Изм.	Лист	№ докум.	Подпись	Дата		

Используемые технологии



Flutter

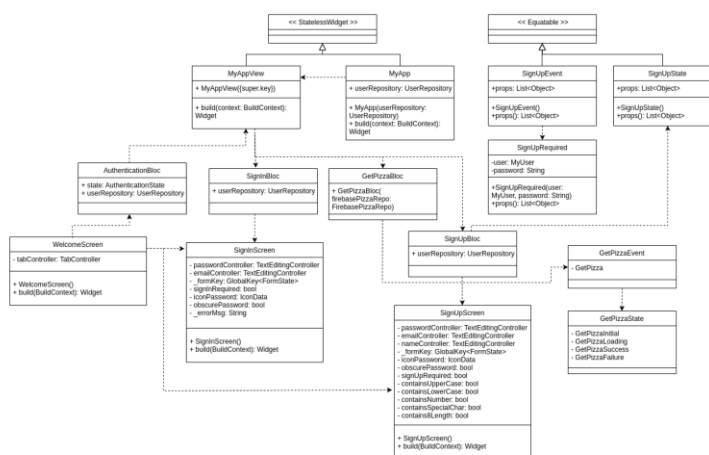


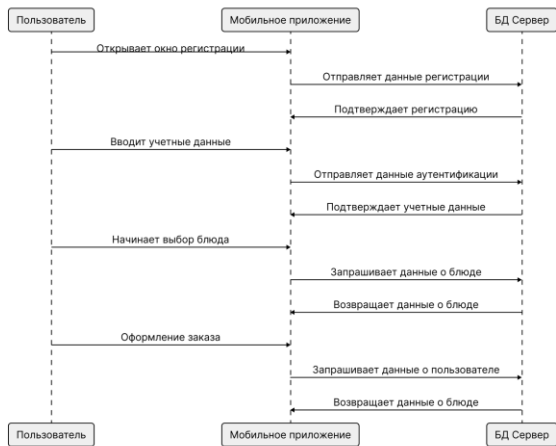
Firebase



Visual Studio Code

Диаграмма классов





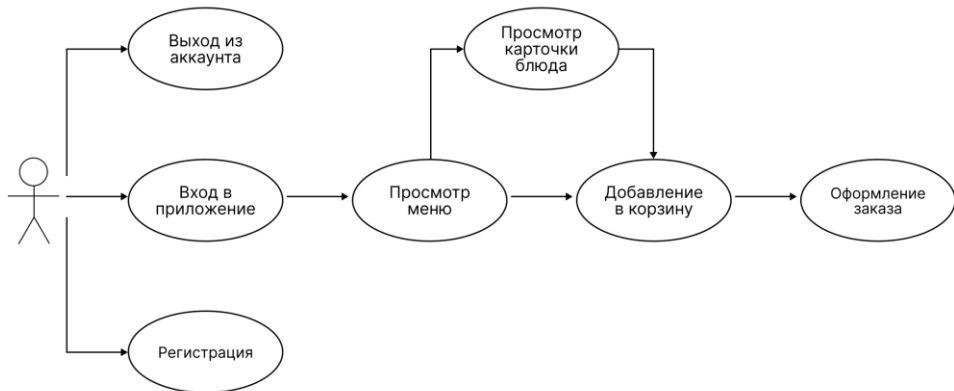
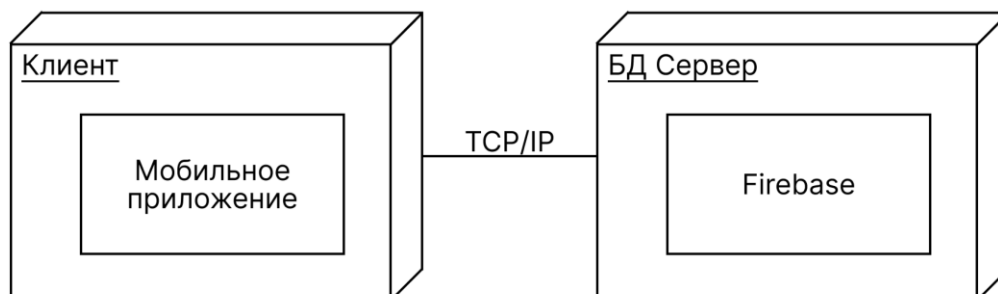
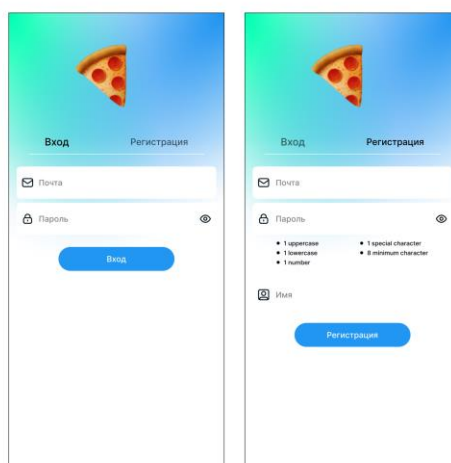


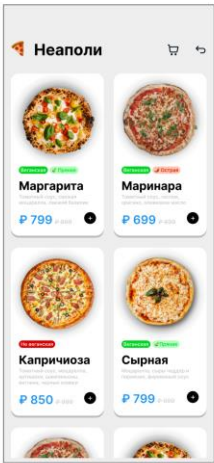
Диаграмма развертывания



Экран входа



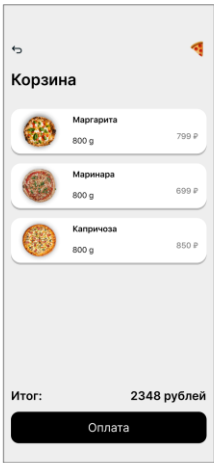
Экран меню



Карточка товара



Экран корзины



Заключение

Спасибо за внимание