Содержание

[Введение 4](#_Toc199244720)

[1 Анализ предметной области и проектирование информационной системы 6](#_Toc199244721)

[1.1 Описание предметной области 6](#_Toc199244722)

[1.2 Построение диаграммы прецедентов 9](#_Toc199244723)

[1.3 Построение диаграммы последовательности 10](#_Toc199244724)

[2 Разработка информационной системы 12](#_Toc199244725)

[2.1 Разработка базы данных 12](#_Toc199244726)

[2.2 Разработка интерфейса пользователя 15](#_Toc199244727)

[2.3 Разработка программного кода 26](#_Toc199244728)

[3 Тестирование информационной системы 41](#_Toc199244729)

[4 Экономическая эффективность внедрения информационной системы 49](#_Toc199244730)

[4.1 Расчет постоянных издержек 49](#_Toc199244731)

[4.2 Расчет переменных издержек 56](#_Toc199244732)

[4.3 Расчет эффективности внедрения 61](#_Toc199244733)

[Заключение 63](#_Toc199244734)

[Литература 64](#_Toc199244735)

[Приложение А Руководство пользователя 65](#_Toc199244736)

[Приложение Б Руководство программиста 69](#_Toc199244737)

[Приложение B Программный код 72](#_Toc199244738)

# Введение

Темой дипломного проекта является «Проектирование и разработка информационной системы для обучения основам графического дизайна».

Целью дипломного проекта – создание мобильного приложения для обучения основам графического дизайна посредством изучения теории и прохождения практических заданий. Были поставлены следующие задачи:

Задачи:

* Анализ предметной области и проектирование информационной системы;
* Построение диаграмм прецедентов, последовательности и «сущность-связь»;
* Разработка базы данных;
* Разработка интерфейса пользователя;
* Разработка программного кода.

Современные технологии и цифровизация оказывают значительное влияние на трансформацию образовательных процессов, затрагивая различные сферы деятельности, включая графический дизайн. В условиях динамичного развития цифровых инструментов, актуализации потребности в квалифицированных специалистах и возрастания требований рынка труда особую значимость приобретает совершенствование методов обучения. Повышение эффективности образовательного процесса требует применения инновационных подходов, сочетающих традиционные методики с возможностями информационных систем, способных интегрировать как теоретические аспекты дисциплины, так и практические навыки, необходимые для профессиональной деятельности.

Развитие современных образовательных платформ предусматривает комплексное использование специализированных программных решений, обеспечивающих доступ к актуальному образовательному контенту, поддержку персонализированного обучения и систематический мониторинг прогресса обучающихся. Внедрение таких решений способствует формированию адаптивных методик, позволяющих модифицировать содержание учебных материалов с учетом уровня подготовки пользователей, а также автоматизировать процесс предоставления обратной связи и анализа усвоенных знаний. Современные образовательные технологии нацелены на создание цифровых инструментов, обеспечивающих эффективное освоение компетенций, востребованных в профессиональной среде.

В контексте подготовки графических дизайнеров особое значение приобретает разработка информационной системы, ориентированного на интеграцию образовательных компонентов в единую цифровую экосистему. Такой подход позволяет не только обеспечить доступ к учебным ресурсам, но и создать условия для углубленного освоения профильных дисциплин, автоматизированного тестирования и совершенствования практических навыков в интерактивной среде. Инструменты цифрового обучения способствуют оптимизации образовательного процесса, повышению доступности и качества, что в свою очередь соответствует современным требованиям подготовки специалистов в сфере графического дизайна.

Формирование эффективной среды для обучения графическому дизайну на основе информационных систем представляет собой актуальное направление, ориентированное на совершенствование профессиональной подготовки и формирование необходимых компетенций. Комплексное использование цифровых технологий, адаптивных образовательных платформ и интерактивных модулей способствует созданию инновационной среды, обеспечивающей высокий уровень усвоения знаний, развитие практических навыков и подготовку специалистов, соответствующих актуальным требованиям отрасли.

Проект направлен на создание цифровой среды, способствующей освоению графических и дизайнерских навыков с высокой степенью доступности и эффективности. Внедрение интерактивного образовательного инструмента позволит обеспечить комплексный подход к изучению теоретических основ и практических приемов графического дизайна, предоставляя пользователям возможность осваивать материал в удобное время и в любом месте.

# 1 Анализ предметной области и проектирование информационной системы

Анализ и проектирование информационной системы для обучения графическому дизайну начинается с определения основных целей и задач системы. Приложение предназначено для комплексного и эффективного обучения графическому дизайну, охватывая как теоретические основы, так и практические навыки. Важным аспектом является предоставление пользователям доступа к разнообразным заданиям и материалам, что способствует развитию базовых и углубленных знаний. Система также включает функционал для мониторинга и оценки прогресса пользователей, что позволяет отслеживать достижения и корректировать образовательный процесс. Это помогает выявить сильные и слабые стороны студентов и направить усилия на дальнейшее совершенствование навыков, повышая квалификацию и профессионализм в области графического дизайна.

## 1.1 Описание предметной области

Графический дизайн представляет собой ключевую дисциплину, которая играет важнейшую роль в современном мире, формируя визуальную коммуникацию и определяя способы взаимодействия людей с информацией. В условиях цифровой трансформации и постоянного роста объемов данных эта область становится неотъемлемой частью множества индустрий, включая маркетинг, издательское дело, разработку программного обеспечения, рекламу и многие другие.

В основе графического дизайна лежит создание эффективных визуальных решений, которые способствуют передаче информации, эмоций и идей через различные медиа-платформы. Это может включать разработку логотипов, фирменных стилей, упаковки, баннеров, интерфейсов сайтов и мобильных приложений, а также создание печатной продукции, такой как книги, журналы и плакаты. Грамотно выполненный дизайн помогает организациям выделяться на конкурентном рынке, повышать узнаваемость бренда.

Приложение для обучения графическому дизайну представляет собой информационную систему, разработанную специально для обеспечения эффективного и комплексного процесса обучения в сфере графического дизайна в рамках дипломного проекта. Данная система сосредоточена на создании, организации и структурировании образовательного контента, охватывающего различные аспекты и направления графического дизайна, включая как теоретические основы, так и практические навыки. Особое внимание уделяется предоставлению доступа к разнообразным заданиям и материалам, направленным на развитие базовых и более углубленных знаний в области графического дизайна. Важным элементом системы является возможность предоставления пользователям доступа к широкому спектру учебных материалов, которые способствуют формированию профессиональных компетенций.

Кроме того, приложение включает в себя функционал для мониторинга и оценки прогресса пользователей, что позволяет отслеживать достижения и своевременно корректировать образовательный процесс. Такой подход способствует более эффективному обучению, помогает выявить сильные и слабые стороны студентов и направить усилия на дальнейшее совершенствование навыков. В результате этого пользователи могут повышать свою квалификацию и достигать высокого уровня профессионализма в области графического дизайна. Система разработана таким образом, чтобы обеспечить максимальную адаптацию под индивидуальные потребности каждого обучающегося, что делает процесс обучения более гибким и результативным.

Приложение предоставляет множество возможностей для обучения через интерактивные курсы, которые включают в себя как теоретические, так и практические задания. Курсы не только помогают повысить уровень знаний в области дизайна, но и способствуют развитию критического мышления, аналитических способностей и творческого подхода к решению задач. Включение практических заданий позволяет применить теоретические знания на практике, что значительно ускоряет процесс усвоения материала и улучшает понимание ключевых принципов графического дизайна. Интерактивные элементы курсов обеспечивают высокий уровень вовлеченности пользователей, что способствует более глубокому изучению материала.

Объектом исследования является приложение для обучения графическому дизайну. Предметом исследования являются процессы, происходящие в системе, такие как:

* + Процесс регистрации пользователей в системе;
  + Процесс формирования и организации теории и заданий;
  + Процесс выполнения практических и теоретических заданий;
  + Процесс мониторинга и оценки прогресса пользователей;
  + Процесс адаптации учебного контента под потребности пользователей;
  + Процесс формирования и предоставления отчетов о достижениях.

В системе предусмотрено хранение данных о пользователях. Для эффективной организации образовательного процесса необходимо собирать информацию, которая включает в себя:

* + Идентификационные данные пользователя;
  + Перечень пройденных курсов и заданий;
  + Результаты выполнения заданий и курсов;
  + Информацию о прогрессе.

Пользователь имеет доступ к уровням, которые будут открываться по мере прохождения. В системе также ведется учет информации о каждом пользователе. О пользователе заносятся следующие данные:

* + Наименование пользователя;
  + Адрес электронной почты;
  + Пароль для доступа в систему;
  + Уникальный идентификатор пользователя;
  + История активности;
  + Прогресс в освоении различных уровней.

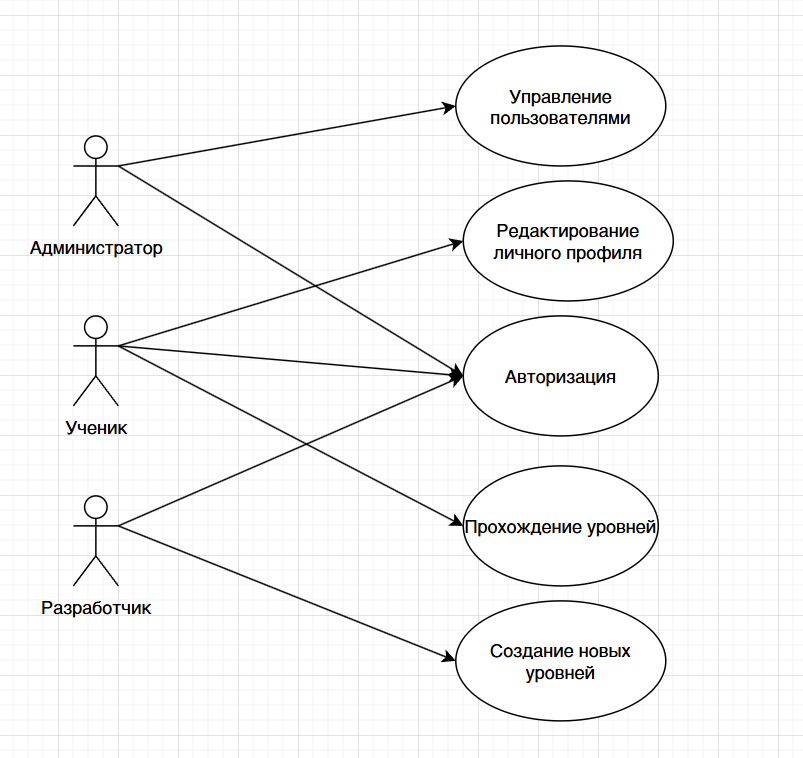
Эти данные позволяют эффективно отслеживать учебный процесс пользователя, адаптировать материалы под уровень и потребности, а также оценивать успехи в обучении. Система разработана таким образом, чтобы обеспечить удобство использования и высокую степень защиты персональных данных.

Таким образом, графический дизайн представляет собой многогранный процесс, сочетающий в себе творчество, технологические навыки и аналитическое мышление. Остается одной из наиболее востребованных областей деятельности, предоставляя специалистам возможности для профессионального роста и внедрения инновационных подходов в создании визуального контента.

## 1.2 Построение диаграммы прецедентов

Диаграмма прецедентов является важным инструментом в процессе проектирования информационных систем, особенно при разработке музыкальных образовательных приложений. Она позволяет визуализировать различные аспекты системы, включая структуру, поведение и взаимодействия между компонентами, что особенно ценно при моделировании сложных процессов музыкального обучения. Применение этого инструмента на ранних этапах разработки помогает выявить потенциальные узкие места в пользовательских сценариях и оптимизировать архитектуру приложения до начала непосредственной реализации.

На диаграмме прецедентов, представленной на рисунке 1.2.1, выделены четыре основные роли пользователей: администратор, ученик и разработчик. Разделение ролей обеспечивает гибкость системы и позволяет адаптировать под нужды различных категорий пользователей. Диаграмма прецедентов также отображает ключевые сценарии взаимодействия, такие как авторизация, выполнение уровней и теории.

Рисунок 1.2.1 – Диаграмма прецедентов

Администратор отвечает за управление пользователями, мониторинг активности и разработку. Это обеспечивает централизованное управление системой и контроль за всеми процессами, что позволяет поддерживать высокий уровень безопасности и эффективности работы системы.

Клиент проходит обучающие уровни, теорию и просматривает личный кабинет. Это обеспечивает персонализированный подход к обучению и возможность отслеживания прогресса, что позволяет пользователям эффективно развивать свои навыки и знания в области графического дизайна.

Разработчик добавляет новые уровней, редактирует имеющиеся уровни и модерирует контент. Это позволяет поддерживать актуальность и качество учебных материалов, а также обеспечивает своевременное обновление контента, что способствует постоянному улучшению образовательного процесса и удовлетворению потребностей пользователей.

## 1.3 Построение диаграммы последовательности

Диаграмма последовательности, представленная на рисунке 1.3.1, является важным инструментом, широко применяемым в процессе проектирования, разработки и анализа различных информационных систем. Ее основная цель заключается в наглядном отображении взаимодействия компонентов системы, а также в детальном представлении последовательности выполняемых действий. Данный визуальный инструмент позволяет структурировать информацию о функционировании системы, что способствует лучшему пониманию логики ее работы и выявлению возможных проблем на ранних этапах разработки.

Использование диаграммы последовательности дает возможность детально проанализировать связи между элементами системы, определить взаимосвязь и последовательность выполнения операций. Это особенно важно в тех случаях, когда требуется точное представление обмена данными между различными объектами и подсистемами. Такой подход позволяет не только выявлять узкие места и потенциальные проблемные участки в функционировании системы, но и разрабатывать стратегии по устранению, что в дальнейшем способствует повышению общей эффективности работы.

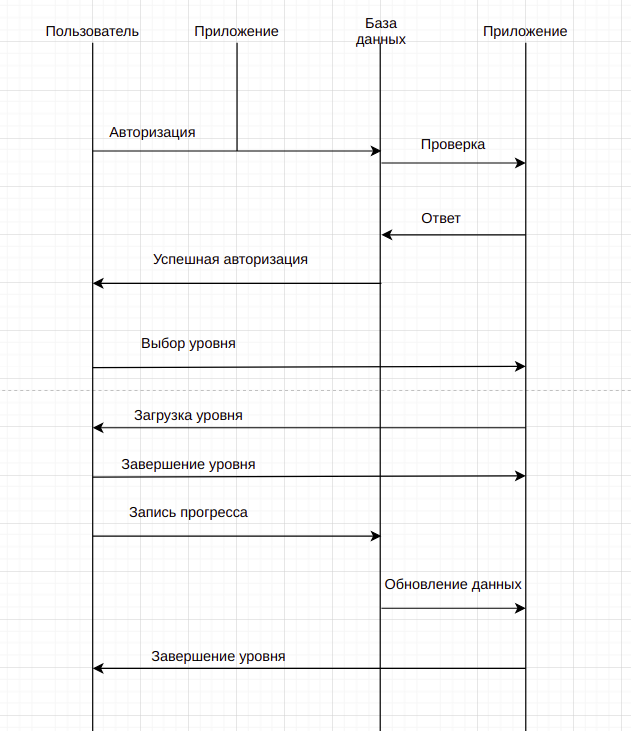


Рисунок 1.3.1 – Диаграмма последовательности

# 2 Разработка информационной системы

Приложение разработано с использованием Unity как основной среды для построения сцены, взаимодействия с объектами и управления пользовательским интерфейсом. Программная логика реализована на языке C#, обеспечивая обработку событий, работу с базой данных, управление сценами и пользовательскими действиями. Firebase применяется для аутентификации пользователей, хранения данных в облаке и синхронизации в реальном времени между клиентскими устройствами, включая реализацию структурированных коллекций, привязанных к уникальному идентификатору пользователя. Визуальное оформление интерфейса и проектирование макетов выполнено в Figma, что позволило систематизировать компоненты, настроить адаптивность и обеспечить единый стиль. Иллюстрации и графические элементы подготовлены в Adobe Illustrator с использованием векторных форматов, что обеспечило чёткость изображения на экранах с высоким разрешением. Полная интеграция этих инструментов позволила сформировать единое цифровое пространство, сочетающее техническую и визуальную составляющие проекта.

## 2.1 Разработка базы данных

Структура базы данных является основой для эффективного хранения и управления данными в приложении. Важность правильной организации базы данных невозможно переоценить, поскольку напрямую влияет на производительность, безопасность и удобство работы системы. Правильная структура позволяет не только обеспечить быстрый доступ к данным, но и эффективно поддерживать целостность и согласованность. Сложная и неэффективно спроектированная база данных может привести к избыточным операциям, проблемам с масштабируемостью и даже к потере данных, что делает проектирование структуры базы данных важным этапом в разработке приложения.

В случае данного приложения для обучения графическому дизайну используется Firebase в качестве решения для хранения данных. Firebase представляет собой облачную платформу, предоставляющую набор инструментов для разработки мобильных и веб-приложений. В отличие от традиционных решений с использованием физических серверов и реляционных баз данных, Firebase предлагает интегрированные сервисы, упрощающие разработку и устраняющие необходимость в ручной настройке серверной инфраструктуры. Применение Firebase позволяет эффективно хранить данные. Рисунок 2.1.1 данные Firebase.

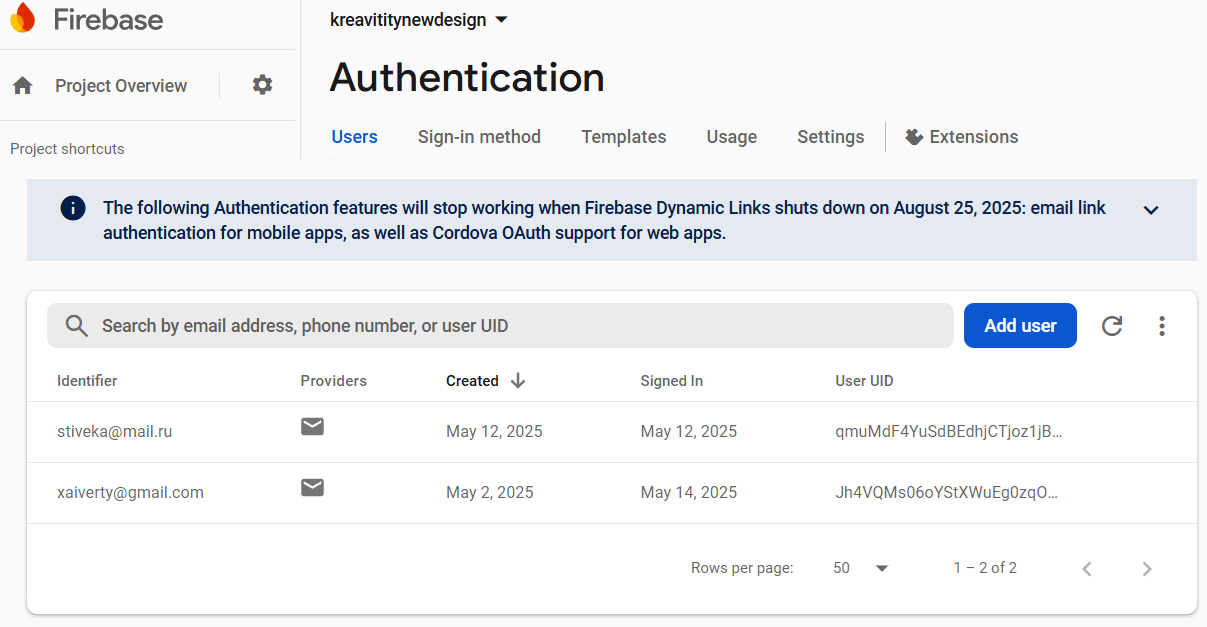


Рисунок 2.1.1 – Данные Firebase

Firebase предоставляет гибкие и масштабируемые решения для хранения данных в реальном времени. Эти технологии позволяют синхронизировать данные между клиентами с минимальной задержкой, обеспечивая мгновенную реакцию интерфейса на любые изменения. Дополнительно реализованы функции аутентификации, что критично для создания персонализированных кабинетов для различных категорий пользователей. Благодаря этому отпадает необходимость в настройке сложной серверной архитектуры, так как все необходимые функции по хранению и обработке данных реализуются средствами Firebase.

Firebase автоматически реализует хэширование критичных данных, таких как пароли пользователей, при использовании встроенной системы аутентификации. Все чувствительные данные обрабатываются с применением современных криптографических алгоритмов, что исключает возможность хранения информации в открытом виде. Процесс хэширования выполняется на стороне сервера и не требует дополнительной настройки, обеспечивая высокий уровень безопасности «из коробки». Это позволяет минимизировать риски, связанные с несанкционированным доступом, и соответствует современным требованиям к защите персональных данных.

В рамках архитектуры информационной системы применяется компонент Firestore — облачное хранилище, поддерживающее структуру коллекций и документов. В Firestore организовано хранение прогресса прохождения образовательных уровней, фиксируется текущее имя участника и накопленные баллы. Такая структура позволяет обеспечить быструю выборку данных и оперативное обновление, что особенно важно для отслеживания результатов в режиме реального времени.

Использование Firebase существенно упрощает задачу управления данными, снижает потребность в поддержке серверной инфраструктуры и сокращает время на развертывание. Это позволяет сосредоточить усилия на реализации образовательного функционала и интерфейсов, минимизируя технические трудности, связанные с обслуживанием базы данных. Данные на рисунке 2.1.2.

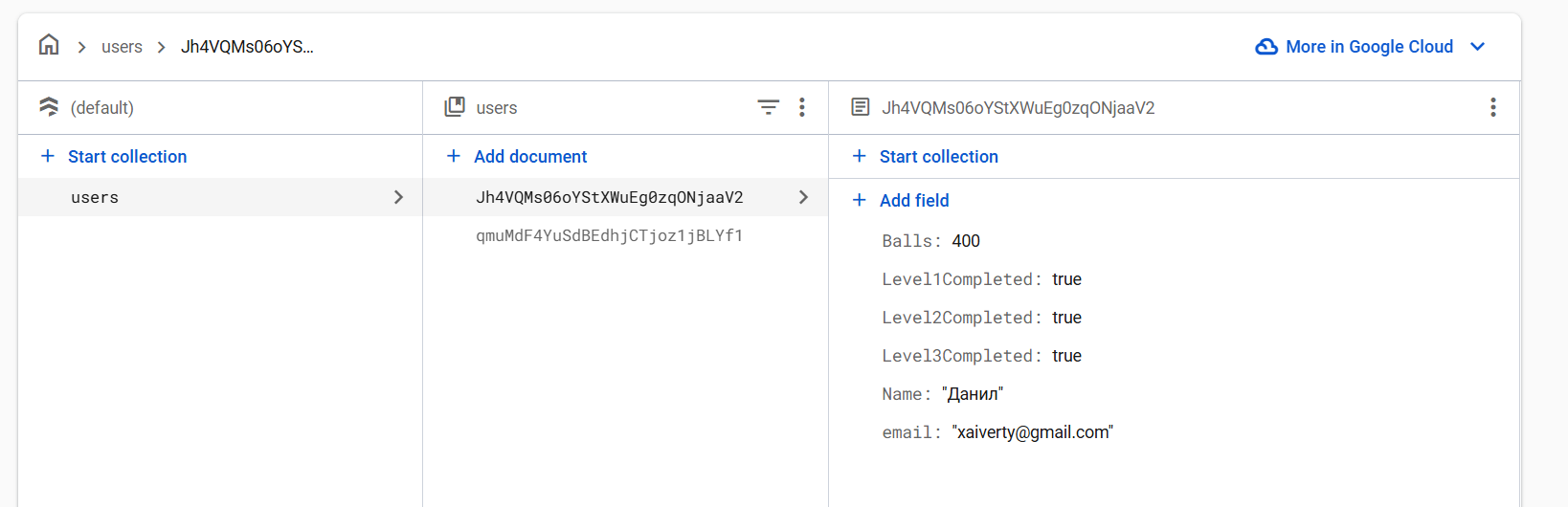


Рисунок 2.1.2 – Данные Firestore

Выбор Firebase в качестве основы для хранения данных является обоснованным решением с учётом специфики приложения, требований к масштабируемости и скорости внедрения. Данное решение обеспечивает требуемую надёжность, безопасность и гибкость, способствуя предоставлению качественного образовательного сервиса.

**2.2 Разработка интерфейса пользователя**

Интерфейс информационной системы играет ключевую роль в успешной работе приложения, поскольку напрямую влияет на восприятие системы пользователем и удобство взаимодействия. Качество интерфейса определяет, насколько интуитивно понятным и доступным будет использование всех функций приложения. Даже самая функциональная система, если имеет сложный или неинтуитивно понятный интерфейс, может вызвать у пользователя трудности и неудовлетворенность, что негативно сказывается на общей эффективности и популярности приложения.

Все элементы интерфейса, включая кнопки, поля ввода, выпадающие списки и информационные панели, разработаны с нуля с учётом специфики задач, решаемых в рамках приложения. Макет интерфейса спроектирован вручную без использования готовых шаблонов, что позволило точно адаптировать структуру и визуальное представление под функциональные и технические требования проекта. Компоненты отрисованы в векторном виде, обеспечена согласованность размеров, отступов и цветовой палитры. Такая организация интерфейса исключает дублирование и обеспечивает целостность пользовательского опыта на всех этапах взаимодействия с системой.

Важно, чтобы интерфейс был не только эстетически привлекательным, но и удобным для различных категорий пользователей. Элементы управления должны быть легко различимы, а навигация — логичной и непрерывной, чтобы пользователь мог быстро и без усилий ориентироваться в приложении. Это способствует сокращению времени на выполнение задач, повышению продуктивности и комфортному использованию приложения. Применение современных принципов UI/UX дизайна гарантирует создание интерфейса, который будет адаптироваться под различные устройства и экраны, обеспечивая удобство работы как на настольных компьютерах, так и на мобильных устройствах.

Спроектированный интерфейс позволяет пользователю сосредоточиться на выполнении основных задач, таких как изучение материалов, выполнение заданий и отслеживание прогресса. Понятный и логичный интерфейс снижает вероятность ошибок и минимизирует необходимость в обучении, поскольку взаимодействие с системой становится естественным и понятным.

Для проектирования интерфейса авторизации и регистрации применены элементы типа Input Field для ввода данных, Label для текстовых пояснений, Canvas в качестве контейнера, объединяющего все визуальные компоненты, и Button для подтверждения действий. Вся структура выстроена с акцентом на функциональность и визуальную согласованность, обеспечивая понятное и интуитивное взаимодействие.

Button запускает процесс авторизации, в котором происходит валидация введённых данных из Input Field, отправка запроса на вход через Firebase, загрузка пользовательских данных из базы и переход к основной сцене при успешной аутентификации. В случае ошибки отображается соответствующее уведомление, а интерфейс возвращается в активное состояние. Код метода на рисунке 2.2.1



Рисунок 2.2.1 – Метод по нажатию кнопки

Интерфейс авторизации должен быть интуитивно понятным и не создавать дополнительных трудностей при регистрации или входе в систему. Необходимо предусмотреть формы для авторизации. Интерфейс на рисунке 2.2.2.

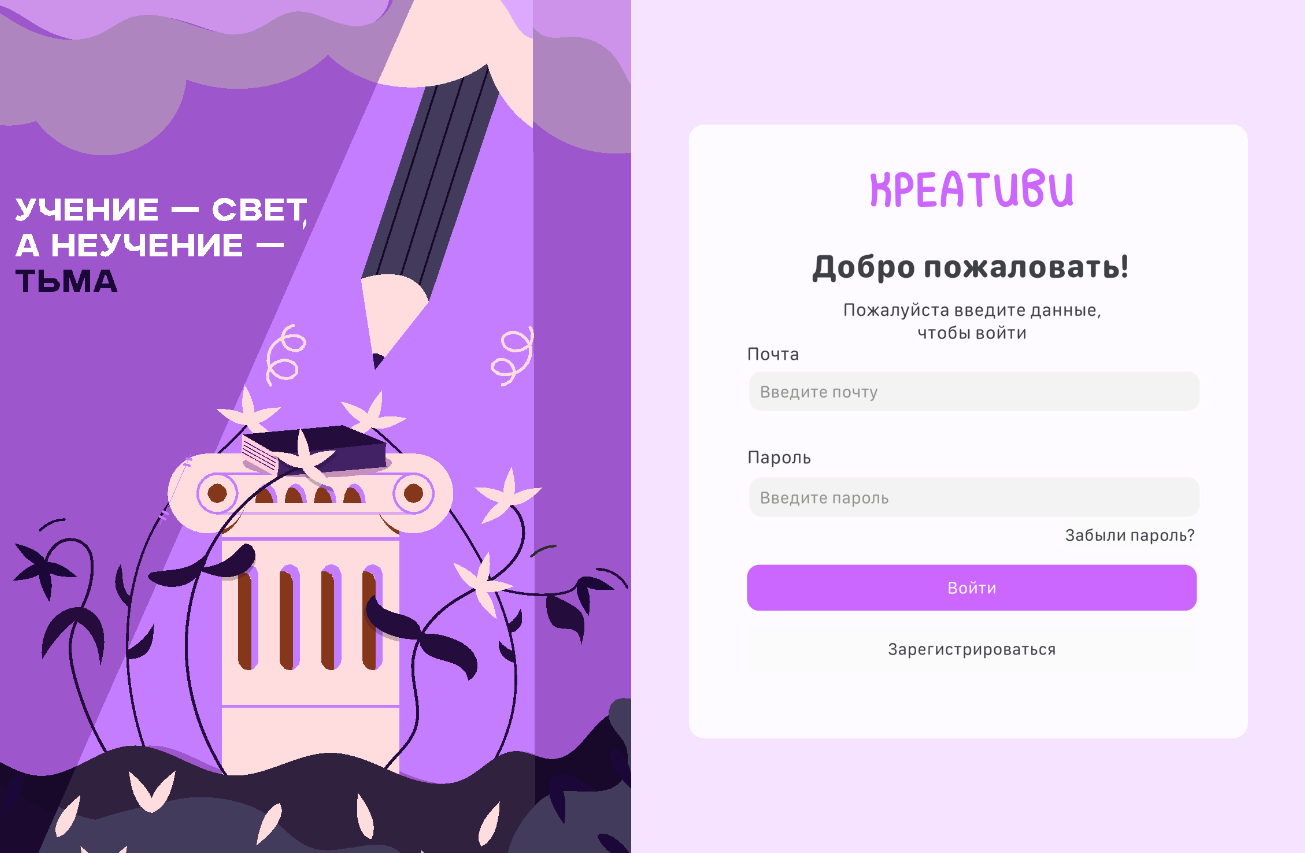


Рисунок 2.2.2 – Интерфейс авторизации

Домашняя страница является важным элементом пользовательского интерфейса, поскольку служит основным ориентиром для пользователей после успешной авторизации. После входа в систему, домашняя страница должна предоставлять пользователю все необходимые инструменты для продолжения работы, обеспечивая быстрый доступ к основным функциям приложения. Это место, где пользователь может увидеть обзор доступных курсов, задач и текущего прогресса, а также найти важную информацию о новостях или обновлениях системы.

На домашней странице задаются GameObject, каждому из которых присваивается визуальное представление с использованием компонента Image, после чего к объектам добавляются обработчики событий, реализующие переходы между сценами и открытие интерфейсных панелей. Назначение слушателей происходит программно, что позволяет динамически управлять логикой навигации и отображения элементов. Код методов на рисунке 2.2.3.



Рисунок 2.2.3 – Код методов домашней страницы

Интерфейс домашней страницы должен быть продуман таким образом, чтобы все ключевые элементы и разделы были легко доступны. Страница должна содержать интуитивно понятные кнопки и ссылки, ведущие к различным разделам приложения, таким как список курсов, задания, результаты и настройки учетной записи. Также на главной странице может быть отображен краткий обзор текущего прогресса пользователя, который будет мотивировать продолжать обучение и выполнять задания. Домашняя страница представлена на рисунках 2.2.4 – 2.2.5.

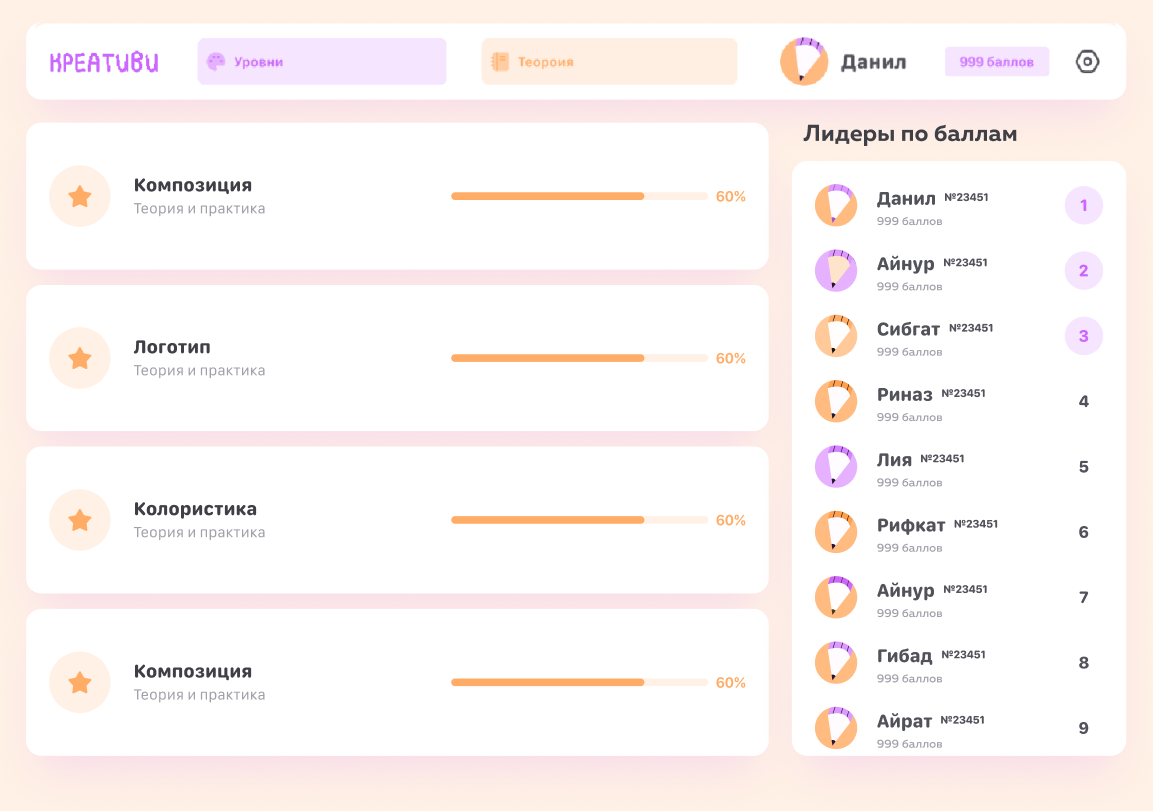


Рисунок 2.2.4 – Интерфейс вкладки уровней

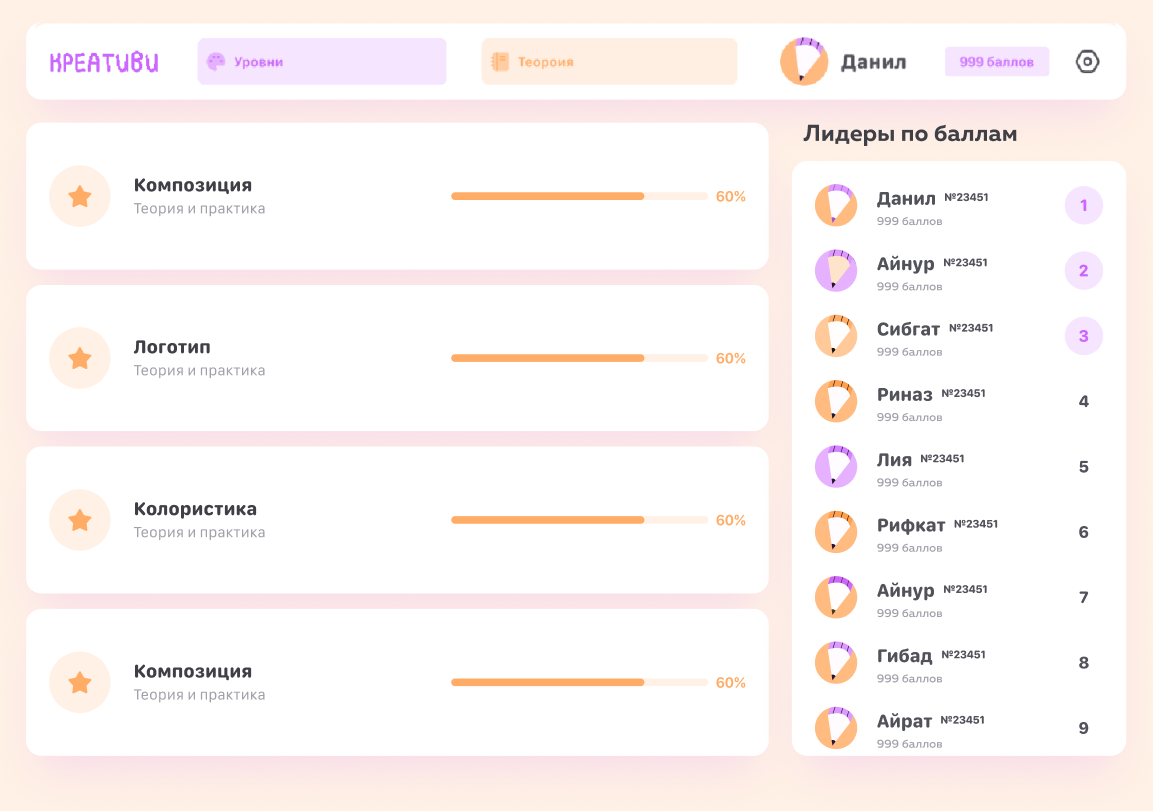


Рисунок 2.2.5 – Интерфейс вкладки теории

Основной задачей интерфейса уровня является представление учебного контента, таких как материалы, задания и инструменты для выполнения заданий, в удобном для восприятия виде. Все элементы должны быть логично организованы и расположены таким образом, чтобы пользователь мог быстро находить необходимые функции. Кнопки и меню должны быть четко видны и легко доступны, а взаимодействие с системой должно происходить без задержек и ошибок. Приятный и лаконичный дизайн интерфейса уровня помогает избежать перегрузки информацией, сохраняя внимание пользователя на основных задачах. Важным аспектом является использование визуальных подсказок и выделение ключевых элементов, таких как кнопки для выполнения заданий. Интерфейс теории на рисунке 2.2.6.

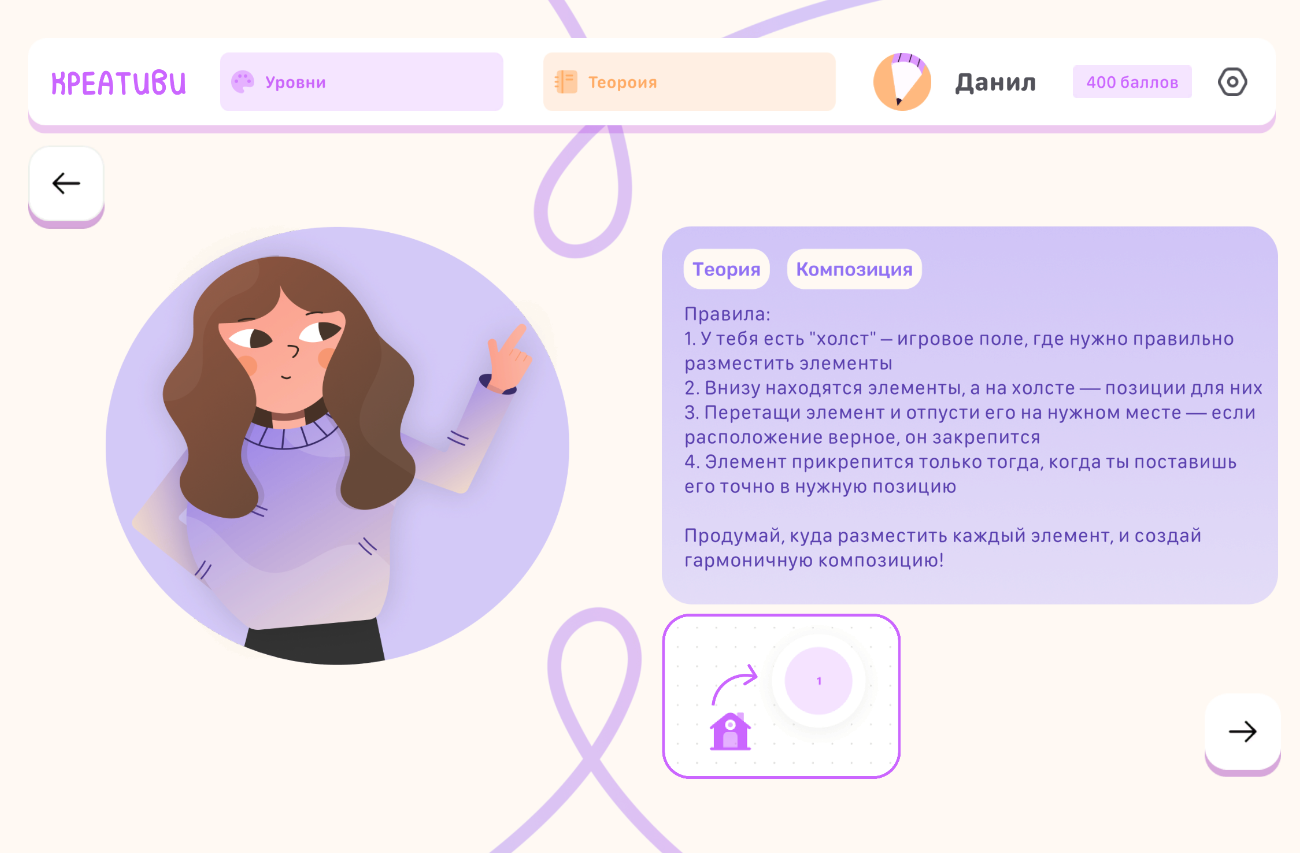


Рисунок 2.2.6 – Интерфейс теории

Интерфейс уровня представляет собой интерактивное пространство, разделённое на несколько ключевых элементов, обеспечивающих понятную и целенаправленную работу пользователя. В верхней или боковой части экрана располагается навигационная панель, содержащая набор иконок, каждая из которых символизирует отдельный объект или элемент, необходимый для выполнения задания.

Основная часть интерфейса отведена под зону действия — область, содержащую определённые поля или зоны, в которые необходимо перенести иконки из панели навигации. Эти зоны заранее обозначены визуально и могут сопровождаться вспомогательными элементами, подсказывающими назначение.

При успешном переносе иконки в соответствующую зону происходит визуальное преобразование: иконка превращается в иллюстрацию, более детализированное изображение или компонент, отражающий значение. Это преобразование сопровождается анимацией, усиливающей эффект завершения действия и создающей визуальную обратную связь.

На уровнях также инициализируются GameObject, визуальные элементы которых оформляются с помощью Image. Поведение этих объектов полностью управляется через программный код, включая обработку событий перетаскивания. В приведённом коде реализуется логика завершения перетаскивания с проверкой попадания в заданную зону, скрытием иконки и зоны, активацией других изображений, запуском анимации и увеличением счётчика действий. Управление осуществляется через компоненты UI и системные вызовы Unity, обеспечивая интерактивность и отслеживание прогресса. Код основного метода уровня на рисунке 2.2.7.



Рисунок 2.2.7 – Метод теории

Процесс повторяется для всех доступных иконок. Задача пользователя — правильно сопоставить и перенести каждую иконку в соответствующую зону. После успешного размещения всех элементов активируется завершающая панель или визуальное уведомление о победе, сигнализирующее об окончании уровня и возможности перехода к следующему этапу. Реализация уровня на рисунке 2.2.8.

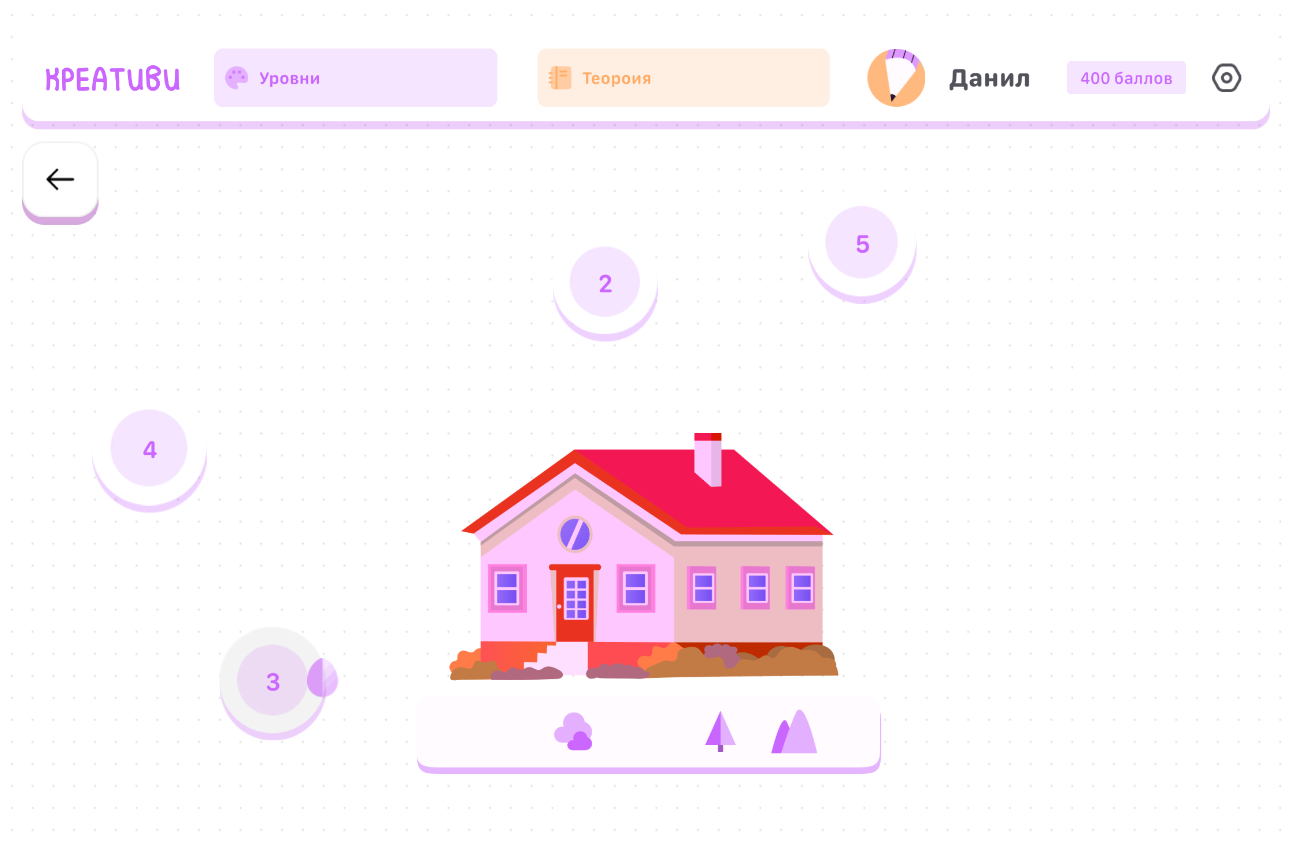


Рисунок 2.2.8 – Интерфейс задания A

Интерфейс уровня организован таким образом, чтобы пользователь мог взаимодействовать с элементами палитры и выполнить задание, связанное с выбором подходящего визуального решения. В верхней или боковой части размещена навигационная панель, содержащая несколько вариантов палитр, каждая из которых представлена в виде перетаскиваемой иконки. Эти палитры различаются по цветовой композиции и визуальному стилю.

В центральной части интерфейса расположена специальная зона — панель выбора, предназначенная для приёма одной из палитр. Пользователь перетаскивает выбранную палитру с панели навигации в эту зону. После переноса палитра масштабируется и закрепляется в панели, визуально обозначая выбор.

Пока палитра размещена, пользователь имеет возможность пересмотреть решение, изменив выбор. После подтверждения с помощью соответствующей кнопки система проверяет правильность выбранной палитры. В случае совпадения с заданным условием отображается финальная панель, сигнализирующая об успешном завершении уровня. Если выбор неверен, пользователь может попробовать снова, заменив палитру на другую.

Такая структура интерфейса способствует формированию осознанного выбора и визуальному обучению, стимулируя анализ и практическое применение полученных знаний. Реализация уровня на рисунке 2.2.9.

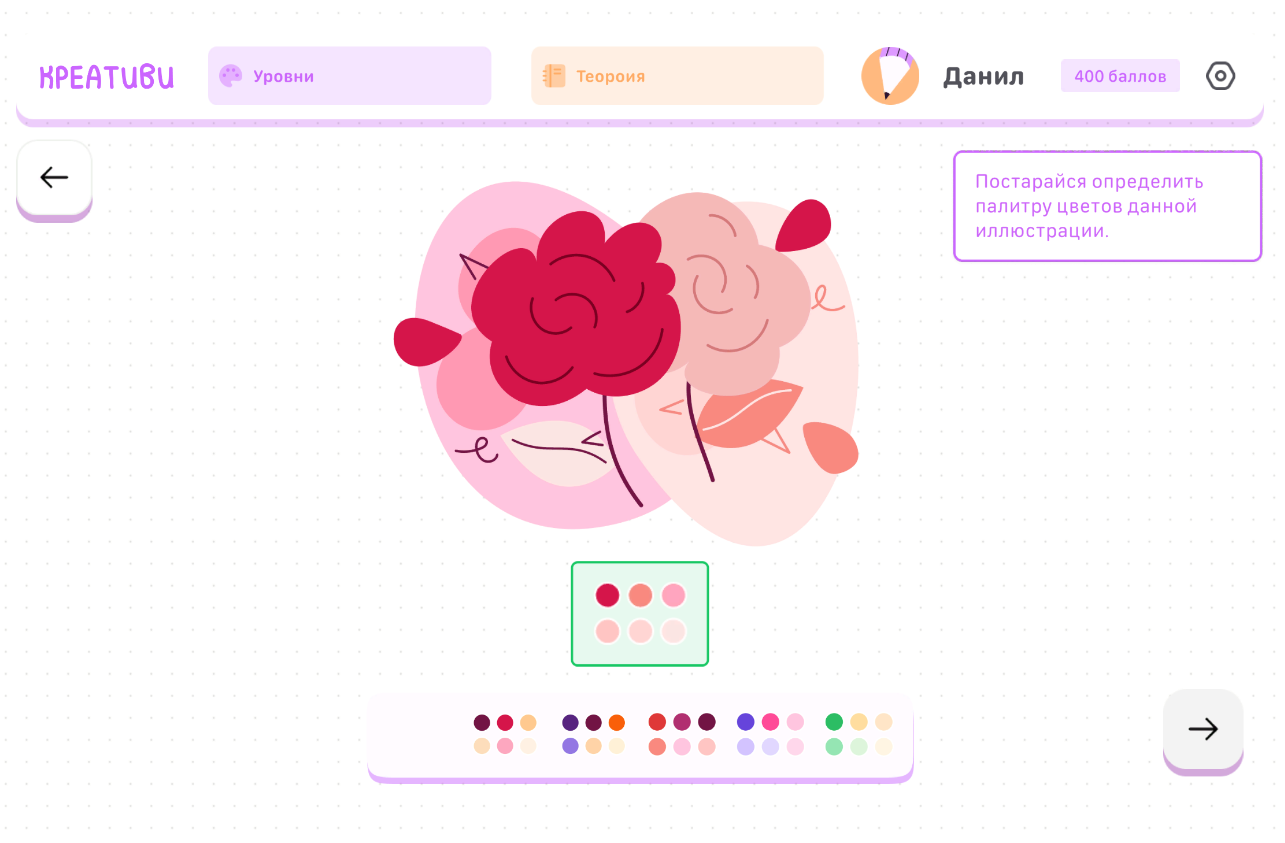
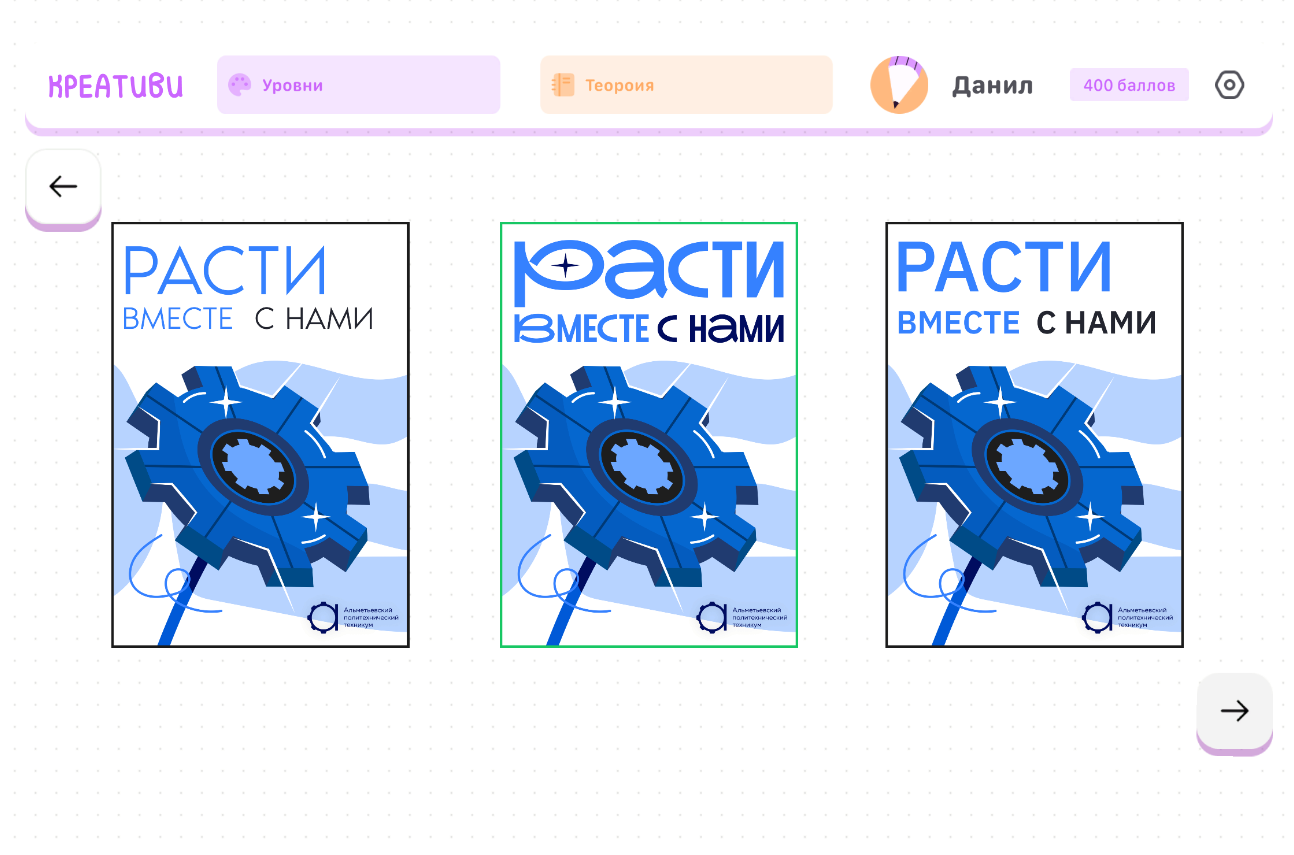


Рисунок 2.2.9 – Интерфейс задания Б

Интерфейс уровня сосредоточен на работе с типографикой и визуальной стилистикой шрифтов. На экране представлены несколько постеров, каждый из которых выполнен с использованием определённого шрифта. Эти постеры расположены в центральной или нижней части интерфейса и служат как варианты для выбора.

Пользователь анализирует представленные постеры и выбирает тот, чей шрифт максимально соответствует заданному стилю. Выбор осуществляется нажатием или перетаскиванием постера в обозначенную зону подтверждения. После этого происходит проверка соответствия: если выбранный шрифт отвечает стилевым требованиям, система подтверждает правильный выбор и отображает панель завершения уровня.

При неверном выборе пользователь может повторить попытку, анализируя другие постеры. Такой подход формирует навык визуального восприятия шрифта, развивает чувство соответствия типографики к стилю дизайна и помогает закрепить знания о различии между шрифтовыми решениями. Реализация уровня на рисунке 2.2.10.

Рисунок 2.2.10 – Интерфейс задания В

**2.3 Разработка программного кода**

Функционал информационной системы играет ключевую роль в обеспечении эффективного обучения и организации работы пользователей внутри системы. Основные функции должны быть направлены на упрощение учебного процесса, повышение доступности материалов и интерактивных заданий, а также на обеспечение удобного мониторинга прогресса. Разработка функционала информационной системы должна учитывать потребности различных типов пользователей, таких как студенты и преподаватели, и обеспечивать поддержку задач в рамках образовательного процесса.

Функционал информационной системы должен включать систему авторизацию, позволяющую пользователям безопасно входить в систему, а также необходимо инициализировать Firebase для подключения к серверу.

Инициализация Firebase в программном модуле необходима для обеспечения надежной и масштабируемой базы данных, а также для реализации множества функций, таких как аутентификация, хранение данных, аналитика и синхронизация данных в реальном времени. Firebase предоставляет облачные решения, которые позволяют организовать хранение данных пользователей, учетных записей, прогресса в обучении и других образовательных материалов, обеспечивая быстрый доступ и безопасность.

Инициализация Firebase важна для настройки и подключения системы к облачной инфраструктуре, что позволяет избежать необходимости развертывания и поддержания отдельного серверного оборудования. Использование Firebase позволяет оптимизировать процесс разработки, так как все необходимые серверные компоненты уже интегрированы в платформу и предоставляют доступ к широкому набору инструментов. Авторизация и инициализация на рисунке 2.3.1.



Рисунок 2.3.1 – Функционал авторизации и инициализации

Функциональные возможности информационной системы формируют основу для обеспечения стабильной и эффективной работы всех компонентов образовательного приложения. Одним из ключевых элементов реализации является подключение к облачной базе данных и организация хранения информации о пользователях и взаимодействии с учебным контентом.

В рамках приложения используется облачное хранилище, основанное на технологии Firestore, предоставляемой платформой Firebase. Этот компонент позволяет хранить структурированные данные с поддержкой вложенных документов, что обеспечивает гибкость при организации информационного пространства. Firestore используется для записи и получения информации о текущем имени пользователя, адресе электронной почты и накопленных баллах. Кроме того, фиксируется прогресс в обучении, что позволяет формировать персонализированные отчёты и адаптировать образовательный маршрут.

Инициализация Firestore осуществляется в момент запуска программного модуля. При этом производится автоматическая проверка всех необходимых зависимостей и подключений. После успешной инициализации активируются механизмы для работы с облачной базой и системой аутентификации. Конкретная реализация предполагает создание глобального экземпляра объекта, который сохраняется при переходе между сценами, что обеспечивает целостность данных в рамках пользовательской сессии.

Функция установки пользовательских данных принимает имя, адрес электронной почты и количество баллов, после чего инициируется процесс сохранения в Firestore. Передача данных осуществляется через словарь, содержащий пары "ключ-значение", где каждый параметр кодируется и передаётся в облачное хранилище. Используется режим слияния, при котором новые данные дополняют уже существующую запись без полного перезаписывания. В результате, при последующих обновлениях сохраняется история и стабильность структуры документа.

При отсутствии соединения с Firestore или при неактивной аутентификации осуществляется проверка на наличие ошибок, что исключает повреждение или потерю данных. После успешной отправки данных выводится подтверждение корректного завершения операции. Использование Firestore в рамках архитектуры приложения обеспечивает устойчивость и масштабируемость, а также упрощает обработку информации о пользователях и результатах.



Рисунок 2.3.2 – Функционал инициализации Firestore

Реализация на основе Firestore позволяет автоматизировать основные операции с данными, обеспечивая синхронность между клиентскими устройствами и облачным хранилищем. Это создаёт прочную основу для персонализации образовательного процесса и повышения эффективности взаимодействия с системой.

Сценарий реализует функциональность перетаскивания объектов с обратной связью и обработкой результата. При старте сохраняется исходная позиция элемента, инициализируются компоненты интерфейса, отключаются вспомогательные изображения, а также скрывается итоговая панель. В процессе перетаскивания объект плавно следует за курсором с заданной скоростью, меняя прозрачность и блокируя взаимодействие с другими элементами интерфейса.

При отпускании объекта выполняется проверка попадания в заданную зону. В случае успеха изображение скрывается, зона деактивируется, активируются связанные изображения, и запускается анимация увеличения и возврата. Также увеличивается счётчик успешно размещённых элементов. После достижения нужного количества объектов запускается корутина с задержкой, по завершении которой отображается финальная панель.

Если объект не попадает в нужную зону, возвращается в исходное положение плавной анимацией. В дополнение реализована обработка выгрузки сцены для сброса счётчика и сохранения корректного состояния при повторной загрузке. Также добавлены кнопки для перехода на другие сцены, назначенные через интерфейс Unity. Код реализации на рисунке 2.3.3.

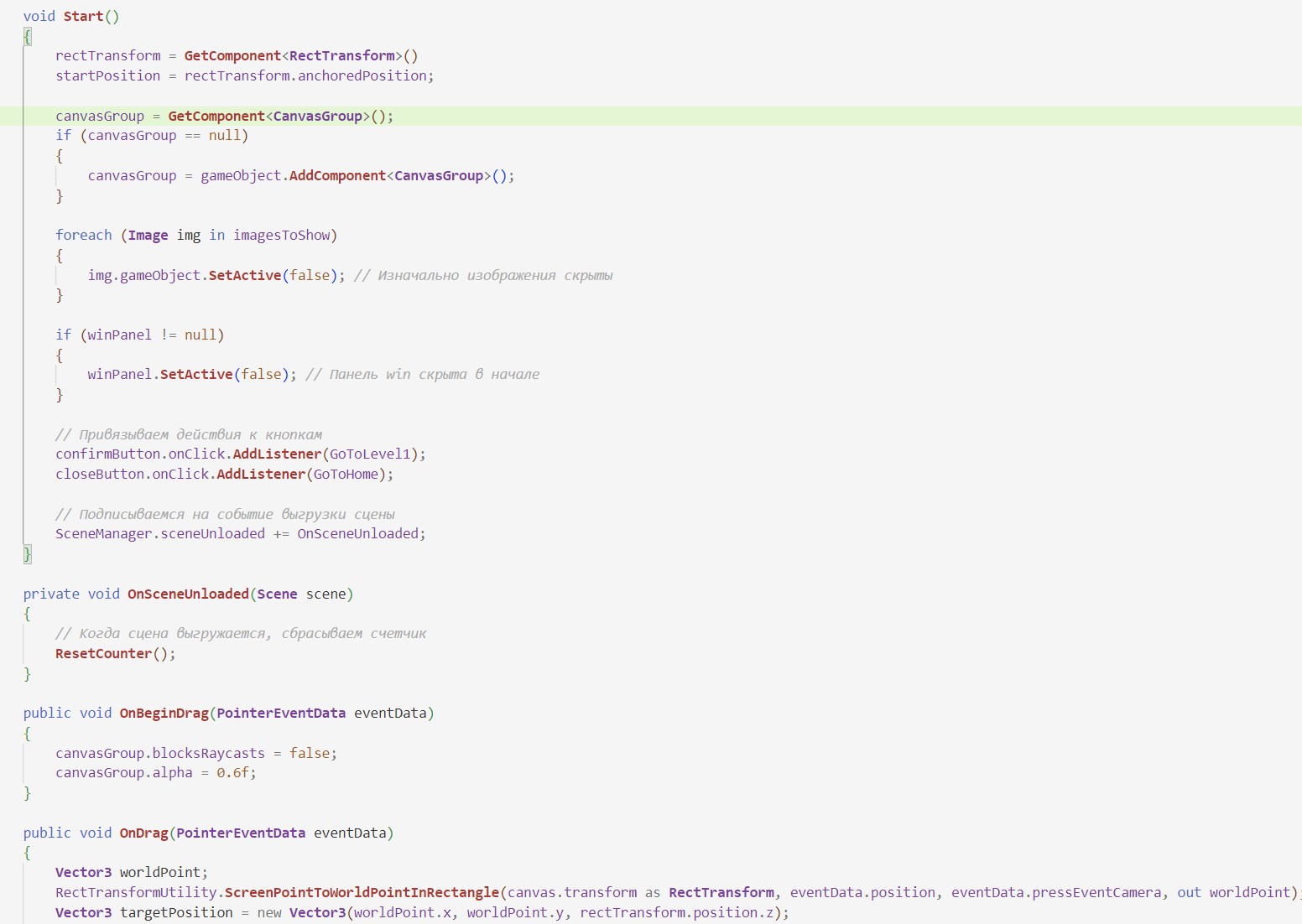


Рисунок 2.3.3 – Функционал размещения объектов

Код реализует простую систему отображения дополнительной информации по нажатию на интерфейсную кнопку. Объект Descriptor, к которому осуществляется ссылка, предназначен для показа описания, подсказки или другого вспомогательного контента. В методе Start объект автоматически скрывается, чтобы избежать преждевременного появления при запуске сцены. Это гарантирует, что пользователь сначала увидит только интерфейс без отвлекающих элементов.

При клике по кнопке, к которой привязан данный компонент, срабатывает метод OnPointerClick, в котором проверяется, неактивен ли Descriptor. Если условие выполнено, объект становится видимым, и запускается функция с задержкой Invoke, по истечении которой Descriptor скрывается через метод HideDescriptor. Это позволяет пользователю кратковременно ознакомиться с информацией, не требуя ручного закрытия окна. Реализация кода на рисунке 2.3.4.



Рисунок 2.3.4 – Функционал отображения интерфейса

Функционал предназначен для управления отображением и анимацией изображений при перетаскивании объектов на определённую вкладку интерфейса. Компонент реализует интерфейсы IPointerEnterHandler и IPointerExitHandler, что позволяет реагировать на наведение и выход курсора мыши, выводя отладочные сообщения для визуального контроля взаимодействия.

В методе Start все изображения из заданного массива скрываются, чтобы не отображались до момента взаимодействия. Основная логика реализуется в методе OnIconDropped, который вызывается при размещении перетаскиваемого объекта на вкладку. При этом активируется соответствующее изображение из массива и запускается анимация.

Функция ShowImageForIcon отвечает за отображение конкретного изображения, связанного с перетащенным объектом, определяемым через метод GetIconIndex. Пока в методе реализована заглушка, возвращающая нулевой индекс — при необходимости здесь может быть добавлена логика идентификации по имени объекта или другому уникальному признаку.

Анимация реализуется через корутину AnimateImage, которая плавно увеличивает масштаб изображения до заданного значения, после чего возвращает к исходному размеру. Между этими фазами предусмотрена пауза, в течение которой изображение сохраняет увеличенное состояние. Функционал на рисунке 2.3.5.

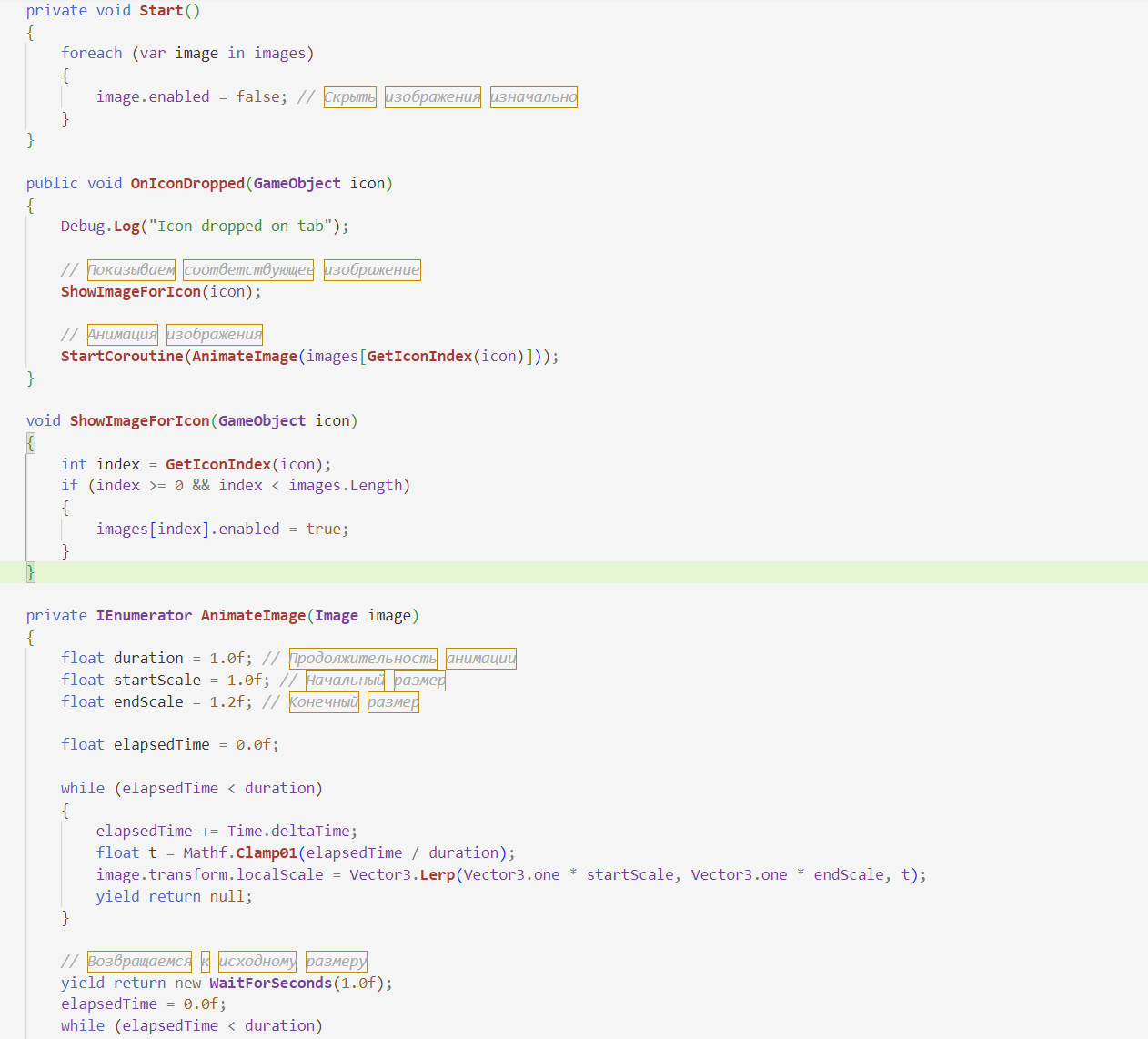


Рисунок 2.3.5 – Функционал управления объектами

Сценарий реализует механику перетаскивания цветовой палитры или другого визуального элемента в интерфейсе Unity. Компонент подключается к объекту, который должен реагировать на действия пользователя, такие как начало, процесс и завершение перетаскивания.

При инициализации сохраняются исходное положение, родительский контейнер и масштаб объекта. Это необходимо для возможности возврата к изначальному состоянию в случае, если перетаскивание завершилось вне допустимой зоны.

В момент начала перетаскивания объект временно переносится на уровень основного Canvas, что позволяет свободно перемещать по экрану, не завися от текущей иерархии. Также временно отключаются блокировки лучей (raycasts), чтобы объект не мешал обработке событий другими элементами интерфейса.

Во время перетаскивания позиция объекта синхронизируется с положением курсора, с учётом преобразования координат экрана в мировые координаты, что обеспечивает точное и плавное следование за мышью.

По завершении перетаскивания проверяется, был ли объект успешно размещён в допустимую зону (TabSlot.lastDroppedOnThisFrame). Если размещение не произошло, вызывается метод ReturnToStart, возвращающий объект в исходное положение, масштаб и иерархию. Также вызывается очистка палитры через отдельный управляющий компонент TabChecker. Функционал на рисунке 2.3.6.



Рисунок 2.3.6 – Механика перетаскивания цветовой палитры

Функционал реализует зону приёма для объектов, поддерживающих перетаскивание — в данном случае, элементов палитры. Компонент подключается к объекту интерфейса и обрабатывает событие OnDrop, происходящее при отпускании перетаскиваемого элемента над зоной.

При срабатывании события определяется, был ли перетащен объект с компонентом PaletteDrag2. Если такой объект найден, осуществляется проверка на наличие уже прикреплённого элемента. В случае, если другая палитра уже размещена в данной зоне, автоматически возвращается в исходное положение.

После этого перетащенный объект прикрепляется к зоне: устанавливается в качестве дочернего объекта, изменяется масштаб (увеличение в 1.5 раза для визуального выделения), а также корректируется позиция внутри контейнера. Объект перемещается на верхний уровень по иерархии, чтобы отображаться поверх других элементов интерфейса.

Сохранение ссылки на текущую палитру (currentPalette) позволяет управлять состоянием в дальнейшем. Передаётся также ссылка на выбранную палитру в управляющий компонент TabChecker22, который может анализировать правильность действий пользователя или использовать эту информацию в логике приложения.

Флаг lastDroppedOnThisFrame используется как глобальный индикатор успешного размещения палитры, предотвращая повторное возвращение объекта в начальную позицию при завершении перетаскивания. Метод ClearPalette обнуляет текущую палитру, позволяя освободить зону и подготовить к новому взаимодействию. Функция на рисунке 2.3.7.



Рисунок 2.3.7 – Зона приёма для объектов

Функционал управляет интерфейсной панелью и навигацией между сценами в приложении. Компонент подключается к различным кнопкам, каждая из которых выполняет определённое действие, связанное с отображением панели и переходами.

При запуске панели устанавливается начальное скрытое состояние. Две отдельные кнопки (extraOpenButton1 и extraOpenButton2) отвечают за активацию панели, предоставляя пользователю доступ к дополнительному контенту или подтверждающей информации.

Кнопка closeButton позволяет закрыть панель, оставляя пользователя на текущей сцене, без каких-либо изменений. Кнопка confirmButton используется для подтверждения действия и инициирует переход на сцену "Home".

Отдельный метод ShowPanelAndGoToLevelTheory1 служит для показа панели с последующим ожиданием закрытия. В нём используется корутина, ожидающая завершения взаимодействия с панелью перед выполнением возможных переходов. В текущей реализации переход на сцену "LevelTheory1" закомментирован, что означает, что фактически сцена не меняется — пользователь остаётся на текущем экране.

Сценарий обеспечивает централизованное управление состоянием панели и связанными переходами, что удобно при реализации диалоговых окон, подтверждений или вспомогательных всплывающих инструкций в обучающих интерфейсах. Функционал на рисунке 2.3.8.



Рисунок 2.3.8 – Управление интерфейсом

Функционал управляет отображением интерфейсной панели и навигацией между сценами в приложении. При запуске панель скрывается, предотвращая преждевременное отображение. Несколько кнопок (openButton, extraOpenButton1, extraOpenButton2) предоставляют пользователю возможность открыть панель из разных частей интерфейса.

Кнопка closeButton, находящаяся внутри самой панели, позволяет закрыть без выхода со сцены, возвращая пользователя к основному экрану.

Кнопка actionButton отвечает за переход на определённую сцену (в данном случае "Home"), а nextSceneButton инициирует переход на сцену "Level2". Названия сцен передаются через параметр метода LoadScene, что делает логику универсальной.

Сценарий обеспечивает простую и централизованную работу с модальными окнами и переходами, позволяя организовать удобную навигацию и взаимодействие с пользователем, особенно в обучающих или демонстрационных интерфейсах. Функционал на рисунке 2.3.9.



Рисунок 2.3.9 – Навигация между сценами

Функционал реализует отображение таблицы лидеров, получая данные из облачного хранилища Firestore. Компонент подключается к массиву UI-элементов LeaderUI, каждый из которых представляет отдельную запись в списке лидеров. Количество отображаемых записей ограничено семью строками.

При запуске сцены инициализируется подключение к базе данных, после чего вызывается метод LoadLeaderboard, выполняющий запрос ко всем документам в коллекции, содержащей данные пользователей. Из каждого документа извлекаются имя и количество набранных баллов. Только записи с положительным значением баллов включаются в список для отображения.

Список пользователей сортируется по убыванию количества баллов, после чего выбираются первые семь результатов. Для каждого UI-элемента вызывается метод SetData, заполняющий данные в таблице, и SetVisible, который управляет отображением строки в зависимости от наличия информации. При отсутствии данных строки скрываются. Реализация данного кода на рисунке 2.3.9.



Рисунок 2.3.9 – Реализация таблицы лидеров

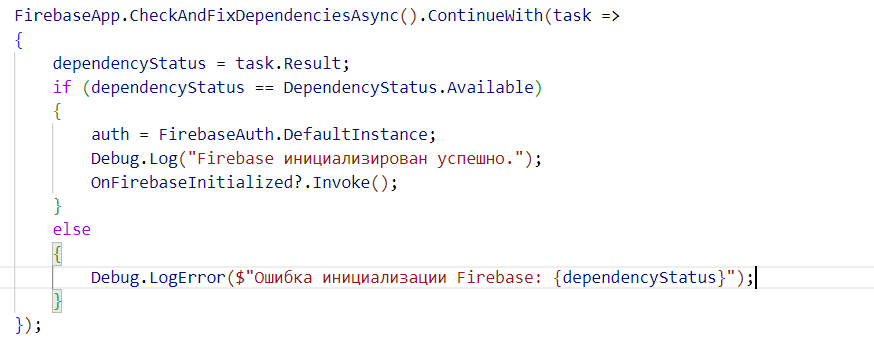
# 3 Тестирование информационной системы

Тестирование информационной системы является важным этапом в процессе разработки, обеспечивающим корректную работу всех функций и взаимодействий внутри системы. На этом этапе проверяется, как информационная система выполняет заявленные задачи и обрабатывает различные сценарии. Тестирование включает в себя проверку логики работы интерфейсов, правильности выполнения действий, таких как перетаскивание объектов, привязка слов к объектам и правильность отображения панелей и кнопок.

В процессе тестирования информационной системы важным инструментом является использование логов, которые позволяют отслеживать и анализировать выполнение программы. Логи фиксируют информацию о действиях, которые происходят в системе, включая ошибки, предупреждения и состояния различных компонентов.

При возникновении ошибок или некорректной работы модулей, логи позволяют точно определить момент сбоя, а также узнать контекст, в котором произошла ошибка. Это облегчает процесс отладки и помогает избежать повторения проблем в будущем. Логи могут содержать информацию о входных данных, параметрах системы, а также сообщениях об исключениях, которые произошли во время выполнения.

Правильно настроенные логи помогают разработчикам оперативно устранять ошибки, улучшая качество информационной системы и повышая стабильность и производительность. Результат лога на рисунках 3.1.

Рисунок 3.1 – Логирование Firebase

Тестирование в Unity представляет собой неотъемлемую часть процесса разработки, способствующую созданию стабильного, надёжного и предсказуемого программного продукта. Оно не только помогает обнаружить и устранить потенциальные ошибки на ранних этапах, но и формирует культуру ответственности за качество кода. Благодаря систематическому подходу к тестированию можно уверенно масштабировать проект, вводить новые функции и изменять существующие компоненты, зная, что ключевые элементы системы защищены проверками.

Юнит-тесты предоставляют возможность изолированно рассматривать отдельные участки логики. Это особенно ценно при работе с критичными алгоритмами, обработкой данных или условиями принятия решений. Тесты фиксируют ожидаемое поведение и служат своеобразной документацией, по которой можно быстро понять назначение и структуру метода. В долгосрочной перспективе это облегчает сопровождение кода и адаптацию новых участников команды.

Интеграционное тестирование охватывает более широкий контекст. Оно подтверждает, что механизмы взаимодействия между системами работают корректно. Это особенно важно в проектах, где множество компонентов работают параллельно или связаны через события, зависимости и совместное использование ресурсов. Такой подход позволяет выявить неочевидные баги, которые проявляются только в динамике взаимодействия.

Проверка пользовательского интерфейса — неотъемлемая часть обеспечения положительного пользовательского опыта. Здесь важна не только техническая стабильность, но и визуальная целостность, отклик на действия, предсказуемость поведения интерфейса. UI-тестирование позволяет убедиться, что взаимодействие происходит так, как задумано: кнопки нажимаются, объекты перетаскиваются, панели открываются и закрываются своевременно. Особенно важно это для проектов, где интерфейс — основной канал общения между системой и пользователем.

Тестирование в Unity охватывает не только технические аспекты, но и организационные процессы внутри команды. Оно способствует выстраиванию прозрачного и предсказуемого цикла разработки, в котором каждая новая функция сопровождается набором проверок. Это снижает риски при внесении изменений и позволяет быстро реагировать на возможные сбои. Регулярное тестирование становится неотъемлемым элементом культуры качества, укрепляя доверие к стабильности проекта как со стороны команды, так и со стороны пользователей.

Внутренний интерфейс Unity предоставляет удобные инструменты для тестирования и отладки, позволяя разработчику напрямую взаимодействовать с объектами сцены, отслеживать состояния в реальном времени и оперативно выявлять ошибки. Через редактор можно запускать сцену в режиме симуляции, наблюдать за выполнением кода, проверять работу скриптов и компонентов без необходимости сборки всего проекта.

С помощью встроенных окон, таких как Console, Inspector, Profiler и Debugger, можно отслеживать сообщения об ошибках, предупреждения, значения переменных, производительность и поведение объектов при различных сценариях. Это значительно упрощает процесс локализации проблем, позволяет быстро находить некорректные участки логики и ускоряет цикл тестирования и исправления.

Таблица 3.1 – Тест-кейс «Проверка на не заполненные поля при авторизации»

|  |  |
| --- | --- |
| Тестовый пример № | 1 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Проверка на заполнение полей при авторизации |
| Краткое изложение теста | Тестирующий пытается авторизоваться, не заполнив поля логина и пароля. |
| Этапы теста | 1. Открытие приложения 2. Нажатие на кнопку «Войти» |
| Тестовые данные | Логин: «Пусто»  Пароль: «Пусто» |
| Ожидаемый результат | Появляется ошибка на заполнение полей. |
| Фактический результат |  |
| Предварительное условие | Поля «Логин» и «Пароль» должны быть пустыми. |
| Статус | Неудачный вход |

Таблица 3.2 – Тест-кейс «Проверка на заполнение полей при авторизации»

|  |  |
| --- | --- |
| Тестовый пример № | 2 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Проверка на заполнение полей при авторизации |
| Краткое изложение теста | Тестирующий пытается авторизоваться, предварительно заполнив логин и пароль |
| Этапы теста | 1.Открытие приложения  2.Нажатие на кнопку «Войти» |
| Тестовые данные | Логин: «xaiverty@gmail.com»  Пароль: «password123@@» |
| Ожидаемый результат | Авторизация прошла успешно |
| Фактический результат |  |
| Предварительное условие | У пользователя есть логин и пароль. |
| Статус | Успешный вход |

Таблица 3.3 – Тест-кейс «Проверка функционала иконок»

|  |  |
| --- | --- |
| Тестовый пример № | 3 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Проверка перетаскивания иконок |
| Краткое изложение теста | Тестирующий пытается перетащить иконки на сцену. |
| Этапы теста | 1.Открытие приложения  2. Нажатие на кнопку «Войти»  3. Открыть уровень  4. Перетащить иконки |
| Ожидаемый результат | Преобразование иконок в объекты |
| Фактический результат |  |
| Предварительное условие | Иконки находятся в исходном положении |
| Статус | Успешное преобразование карты в объект композиции |

Таблица 3.4 – Тест-кейс «Проверка функционала карт»

|  |  |
| --- | --- |
| Тестовый пример № | 4 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Проверка перетаскивания палитры |
| Краткое изложение теста | Тестирующий пытается перетащить палитру на сцену. |
| Этапы теста | 1.Открытие приложения  2. Нажатие на кнопку «Войти»  3. Открыть уровень  4. Перетащить палитру |
| Ожидаемый результат | Преобразование палитру в панель |
| Фактический результат |  |
| Предварительное условие | Иконки находятся в исходном положении |
| Статус | Корректное прикрепление палитры к заданной области |

Таблица 3.5 – Юнит-тест «Проверка прохождения уровня с ошибками»

|  |  |
| --- | --- |
| Тестовый пример № | 5 |
| Приоритет тестирования | Низкий |
| Заголовок/название теста | Нажатие на кнопку на форме авторизации |
| Краткое изложение теста | Тестирующий нажимает на кнопку |
| Этапы теста | 1. Открытие приложения  2. Нажатие на кнопку «Авторизоваться» |
| Ожидаемый результат | Кнопка имеет режим наведения |
| Фактический результат | Наведение срабатывает |
| Предварительное условие | Пользователь нажимает на кнопку |
| Статус | Корректное поведение кнопки |

Таблица 3.6 – Юнит-тест «Проверка перехода между сценами»

|  |  |
| --- | --- |
| Тестовый пример № | 6 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Нажатие на кнопку на форме авторизации |
| Краткое изложение теста | Тестирующий нажимает на кнопку |
| Этапы теста | 1. Открытие приложения  2. Нажатие на кнопку «Авторизоваться»  3. Переход на следующую сцену |
| Ожидаемый результат | Переход на сцену |
| Фактический результат | Переход срабатывает |
| Предварительное условие | Пользователь нажимает на кнопку |
| Статус | Корректное поведение адресации |

# 4 Экономическая эффективность внедрения информационной системы

## 4.1 Расчет постоянных издержек

Применение современных информационных технологий сопряжено с капитальными затратами на приобретение вычислительной техники, на разработку программных продуктов и их внедрение в управленческо-производственный процесс, обучение и подготовку персонала.

Эффективность является сложной экономической категорией, которая складывается на предприятии под влиянием множества факторов: экономических, социальных, правовых и других.

В экономической и научно-технической литературе термин «эффективность» понимается по-разному:

* Как вероятность выполнения поставленных перед системой задач;
* Как отношение реализованного эффекта к максимально возможному.

Экономическую эффективность лучше определять, как меру целесообразности проведения тех или иных мероприятий и выражать ее количественными величинами. Кроме того, под экономической эффективностью обычно понимают отношение между полученными результатами и затратами средств и труда.

Тогда можно дать и такое определение экономической эффективности — это способ действий, обеспечивающий получение в результате осуществляемых усилий и затрат ресурсов максимального (наилучшего) результата.

Понятие эффективности предполагает оценку результатов функционирования системы; это показатель, сопоставляющий в той или иной форме результаты функционирования системы. Общей конечной целью в данном случае является улучшение деятельности предприятия. Частными целями могут быть: снижение затрат на обработку информации; сокращение времени получения результатной информации; получение новой информации, которую без применения ЭВМ получить невозможно.

В современных условиях применяется более широкое понятие социально-экономической эффективности, включающее категории социальных издержек (заболеваемость, загрязнение окружающей среды и т.д.) и социальных благ (здоровье, научный потенциал).

Чтобы выявить в расчетах экономическую эффективность, надо знать, в каких показателях могут быть выражены результаты внедрения информационной системы. Оценка экономической эффективности состоит в определении ряда показателей, характеризующих использование различных видов ресурсов: повышение качества и снижение себестоимости, рост производительности труда управленческих работников и других. Одни из показателей дают оценку прямого эффекта от применения программных продуктов, другие косвенно характеризуют экономическую эффективность.

Общая экономическая эффективность слагается из прямой и косвенной эффективности. Прямая эффективность связана с сокращением затрат труда, с экономией материально-трудовых ресурсов и денежных средств, полученной в результате уменьшения численности персонала, расхода основных средств и вспомогательных материалов. Показатели ее могут быть измерены и выражены в количественных величинах.

Косвенная эффективность проявляется в улучшении работы управленческого персонала, благодаря использованию всесторонней и более качественной информации, что отражается на конечных результатах финансово-хозяйственной деятельности предприятия. Ее критериями могут быть:

* Сокращение сроков формирования документов;
* Повышение качества планово-учетных, контрольных и аналитических операций;
* Сокращение объема документооборота;
* Повышение управленческой культуры;
* Рост производительности труда.

Основным показателем является повышение качества управления, которое, как и при прямой эффективности, ведет к экономии живого и овеществленного труда.

Показатели прямой эффективности подразделяются на первичные и производные и могут быть трудовыми и стоимостными. К таким показателям относят следующие:

* Экономия рабочего времени;
* Индекс экономии затрат труда;
* Индекс производительности труда;
* Коэффициент снижения себестоимости и др.

Обобщающими показателями прямой эффективности, позволяющими судить о целесообразности применения технических средств, являются:

* Годовой экономический эффект;
* Период окупаемости единовременных затрат;
* Коэффициент экономической эффективности (рентабельности) затрат или капитальных вложений.

Экономический эффект – это результат внедрения какого-либо мероприятия, выраженный в стоимостной форме, в виде экономии от осуществления.

Коэффициент экономической эффективности капитальных вложений показывает величину годового прироста прибыли, образующуюся в результате эксплуатации программного изделия, на один рубль капитальных единовременных вложений.

Период окупаемости (величина обратная коэффициенту эффективности) представляет собой период времени, в течение которого произведенные затраты на программное изделие окупаются полученным эффектом.

Определение эффективности информационной системы основано на принципах экономической оценки производства и использования новой техники.

На различных стадиях жизненного цикла информационной системы и в зависимости от целей расчета рассчитываются и документально оформляются следующие виды экономического эффекта: предварительный; потенциальный; гарантированный; фактический.

Предварительный экономический эффект рассчитывается до выполнения разработки на основе данных технических предложений и прогноза использования.

Потенциальный экономический эффект рассчитывается на основе достигнутых технико-экономических характеристик разработанной информационной системы. Потенциальный эффект используется при оценке деятельности организаций-разработчиков информационных систем.

Гарантированный экономический эффект рассчитывается в виде эффекта для конкретного объекта внедрения и общего гарантированного внедрения по ряду объектов.

Фактический экономический эффект рассчитывается на основе данных учета и сопоставления затрат и результатов при конкретных применениях информационной системы.

Количественно измерить влияние автоматизированной обработки данных на результаты финансово-хозяйственной деятельности не всегда возможно, так как повышение эффективности зависит не только от применения технических средств, но и от влияния других многочисленных факторов.

Перечень работ по созданию автоматизированной информационной системы приведен в таблице 4.1.1.

Таблица 4.1.1 – Перечень работ по стадиям проектирования программного модуля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование этапа | Продолжительность  (трудоемкость, чел\*час) | Удельный вес,  % | Исполнители | |
| Руководитель | Программист |
| Проектирование информационной системы | 30 | 7 | + | + |
| Разработка информационной системы | 380 | 87 | - | + |

Продолжение таблицы 4.1.2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тестирование информационной системы | 20 | 5 | - | + |
| Оформление документов | 10 | 3 | + | \_ |
| Итог | 440 | 102 | 2 | 3 |

График 4.1.1 – Линейный график разработки АИС:

В результате составленной таблицы наибольшее количество трудозатрат при разработке программного продукта приходится на разработку информационной системы.

Для оценки экономического эффекта от внедрения мобильного приложения в образовательной организации необходимо рассчитать затраты на разработку.

Величина затрат на создание приложения определяется на основе метода калькуляций по отдельным статьям расходов и их последующим суммированием.

Разработка приложения состоит из нескольких стадий проектирования. В качестве разработчиков системы выступают: руководитель и программист (студент). Таким образом, руководитель участвует 50%, а программист 100% в проекте. Программист принимается за единицу приведенного исполнителя, следовательно, руководитель составляет 0,5 приведенного исполнителя. Т.е. в разработке проекта участвует 1,5 приведенного человека.

Трудоемкость = 440/1,5 = 294 чел ∗ час

Срок исполнения, учитывая, что рабочий день составляет 8 ч., получается 55 дней работы над проектом.

В реальности длительность работы над проектом больше примерно на 30%, поэтому можно принять за срок разработки 72 дней.

Постоянные издержки включают в себя амортизационные отчисления на компьютер и программное обеспечение и затраты на текущий ремонт.

Амортизационные отчисления на компьютер и программное обеспечение производятся ускоренным методом с тем условием, что срок морального старения происходит через четыре года. Таблица представлена на 4.1.2.

Таблица 4.1.2 – Перечень используемого оборудования

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование изделия | Количество,  шт. | Цена за единицу,  руб/шт. | Сумма затрат,  шт. |
| Ноутбук | 1 | 185000 | 185000 |
| Мышь | 1 | 7000 | 7000 |
| Принтер | 1 | 20000 | 20000 |
| Итог | | | 212000 |

Балансовая стоимость ЭВМ включает отпускную цену, расходы на транспортировку, монтаж оборудования и наладку и вычисляется по формуле:

(1)

где,     – первоначальная стоимость оборудования, руб.;

  – норматив амортизационных отчислений на полное восстановление, % (20%);

  – количество месяцев работы оборудования при создании АИС.

Из таблицы 2 видно, что первоначальная стоимость данного оборудования составляет 212000 руб. Оборудование использовалось 440 часов (160 часов в месяц) или ~2,8 месяца. Рассчитаем амортизационные отчисления оборудования:

Аоб = (212000 \* 0,2 \* 2,8) / 12 = 9893,33 руб.

Остальные программные средства, используемые для разработки, предоставляются бесплатно.

Затраты на текущий ремонт и обслуживание оборудования рассчитываются по формуле:

(2)

где,  https://konspekta.net/megaobuchalkaru/imgbaza/baza14/8190227292689.files/image056.gif  – первоначальная стоимость оборудования, руб.;

https://konspekta.net/megaobuchalkaru/imgbaza/baza14/8190227292689.files/image058.gif  – норматив отчислений на ремонт и обслуживание оборудования, % (5%);

https://konspekta.net/megaobuchalkaru/imgbaza/baza14/8190227292689.files/image060.gif  – количество месяцев работы оборудования при создании АИС:

Рассчитаем затраты на текущий ремонт и обслуживание оборудования:

Зро = (212000 \* 0,05 \* 2,8) / 12 = 2473,33 руб.

Полученные данные составляют постоянные издержки и приведены в таблице. Большую долю в постоянных издержках занимают амортизационные отчисления на используемый компьютер и программное обеспечение. Таблица представлена на 4.1.3.

Таблица 4.1.3 – Постоянные издержки

|  |  |  |
| --- | --- | --- |
| Вид постоянных издержек | Затраты, руб. | Удельный вес, % |
| Амортизационные отчисления | 9893,33 | 80 |
| Текущий ремонт | 2473,33 | 20 |
| Итого | 12366,66 | 100 |

## 4.2 Расчет переменных издержек

Переменные издержки включают в себя:

* Материальные затраты;
* Затраты на потребляемую электроэнергию;
* Затраты на оплату труда;
* Отчисления по налогам.

Материальные затраты

К материальным затратам относятся отчисления на материалы, которые используются в процессе разработки и внедрения настольного приложения. Использованные для работы материалы представлены в таблице 4.2.1.

Таблица 4.2.1 – Материальные расходы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование изделия | Количество, шт. | Цена за единицу, руб./шт. | Сумма затрат, руб. |
| Adobe Creative Cloud | 12 | 1200 | 14400 |
| Интернет | 3 | 600 | 1800 |
| Картриджи принтера | 1 | 500 | 500 |
| Бумага А4 | 2 | 250 | 500 |
| Figma | 12 | 1000 | 12000 |
| Итог | | | 29200 |

К затратам на электроэнергию относится стоимость потребляемой электроэнергии ЭВМ за период разработки.

Затраты на электроэнергию рассчитываются по формуле:

(3)

где,    – норма расхода энергии в единицу времени i-го потребителя, кВт/ч;

  – время работы i-го оборудования при создании АИС, ч.;

  – цена 1 кВт/ч энергии, руб. (по данным на 2025 г. стоимость 1кВт/ч для населения составляет 5,85, данные брались с сайта https://tatenergosbyt.ru/supply/tarifs);

  – общее количество потребителей энергии.

Расчет затрат на электроэнергию сведен в таблице 4.2.2.

Таблица 4.2.2 – Затраты на электроэнергию

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Потребитель | Норма расхода энергии, кВт/ч. | Время работы, ч. | Цена 1 кВт/ч, руб. | Сумма затрат, руб. |
| Ноутбук | 0,52 | 440 | 5,85 | 1338,48 |
| Принтер | 0,5 | 6 | 5,85 | 17,55 |
| Роутер | 0,1 | 440 | 5,85 | 257,4 |
| Итог | | | | 1613,43 |

Расчет основной заработной платы производится на основе доли выполнения работы и величины месячного должностного оклада разработчика.

При расчете основной заработной платы, представленном в таблице 4.2.3, за период разработки необходимо учесть, что руководитель участвует в разработке проекта только на этапах системного анализа и анализа требований, которые занимают 10% всего времени, оклад куратора 41000.

Затраты на оплату труда разработчиков информационной системы включают в себя основную заработную плату разработчика.

Для этого нам необходимо применить следующую формулу:

(4)

где,  – часовая тарифная ставка с учетом доплат, руб./ч;

  – трудоемкость выполнения i-го вида работ при создании АИС, ч.

Зо = 440 \* 300 = 132000 руб.

Дополнительная заработная определяется по формуле:

(5)

где,   – норматив дополнительной заработной платы, % (9%);

Зо – основная заработная плата работника.

Рассчитаем затраты на дополнительную заработную плату:

Зд = 132000\* 0,1 = 13200 (руб.) (программист)

Зд = 41000\* 0,1 = 4100 (руб.) (куратор)

Расчет отчислений на социальное страхование ведется по формуле:

(6)

где,   – норматив отчислений на социальное страхование, (30%);

  – фонд заработной платы, руб.

Рассчитаем размер отчислений на социальное страхование:

Зсс = (132000+ 13200) \* 0,3 = 43560 (руб.) (программист)

Зсс = (41000 + 4100) \* 0,3 = 13530 (руб.) (куратор)

Таблица 4.2.3 – Расчет фонда заработной платы за период разработки

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Должность | Вид заработной платы (руб.) | | Сумма, руб. | ФСС, руб. | Удельный  вес, % |
| ЗП осн. | ЗП доп. |
| Программист, руб. | 132000 | 43560 | 175560 | 43560 | 75,19 |
| Куратор, руб. | 41000 | 13530 | 54530 | 13530 | 24,81 |
| Итого | 173000 | 57090 | 230090 | 57090 | 100 |

Накладные расходы рассчитываются по формуле:

(7)

где, – норматив накладных расходов, (50%).

Рассчитаем затраты на накладные расходы:

Рн = 132000\* 0,5 = 66000руб.

Таблица 4.2.4 – Переменные издержки

|  |  |  |
| --- | --- | --- |
| Вид переменных издержек | Величина, руб. | Удельный вес, % |
| Материальные затраты | 29200 | 6,49 |
| Затраты на электроэнергию | 1613,43 | 0,36 |
| Затраты на оплату труда | 230090 | 51,12 |
| Отчисления по налогам | 57090 | 12,7 |
| Накладные расходы | 132000 | 29,33 |
| Итого | 449993,43 | 100 |

Далее необходимо рассчитать общие затраты. На эту статью относятся все издержки, которые были произведены при создании настольного приложения.

Полная себестоимость разработки определяется суммированием постоянных и переменных издержек и вычисляется по формуле:

(8)

где, *Зоб* – себестоимость приложения;

З*пос* – постоянные издержки;

*Зпер* – переменные издержки.

Таким образом, себестоимость создаваемого программного модуля равна:

Зоб = 12366,66 + 462360,09 = 474726,75 руб.

В таблице 4.2.5 приведена структура полных издержек. В результате, при создании программного продукта наибольший удельный вес занимают переменные издержки.

Таблица 4.2.5 – Структура полных издержек

|  |  |  |
| --- | --- | --- |
| Вид издержек | Величина, руб. | Удельный вес, % |
| Постоянные | 12366,66 | 2,6 |
| Переменные | 462360,09 | 97,4 |
| Итого | 474726,75 | 100 |

В таблице 4.2.6 показано процентное соотношение всех категорий затрат на разработку программного модуля.

Таблица 4.2.6 – Соотношение категорий затрат на разработку программного модуля

Продолжение таблицы 4.2.6.

|  |  |  |
| --- | --- | --- |
| Вид затрат | Величина, руб. | Удельный вес, % |
| Затраты на оплату труда | 230090 | 49,77 |
| Отчисления по налогам | 57090 | 12,36 |
| Амортизационные отчисления | 132000 | 28,55 |
| Текущий ремонт | 9893,33 | 2,14 |
| Материальные затраты | 29200 | 6,31 |
| Затраты на электроэнергию | 1613,43 | 0,34 |
| Накладные расходы | 2473,33 | 0,53 |
| Итого | 462360,09 | 100 |

## 4.3 Расчет эффективности внедрения

Разработка программного модуля для обучения основам графического дизайна способствует ускоренному освоению ключевых навыков, улучшая восприятие визуальных элементов, композиции и цветовых решений, что приводит к более быстрому прогрессу в изучении основ дизайна. Упрощение и автоматизация процесса обучения позволяют сократить время на освоение материала на 20%, повышая общую эффективность работы и открывая новые возможности для профессионального развития.

Для расчета точки безубыточности используется формула:

(9)

Подставив необходимые значения, был получен результат:

Экономическая эффективность приложения определяется способностью быстро окупить вложенные в разработку средства. При стоимости подписки 300 рублей в месяц и затратах на разработку в размере 474726,75 рублей, точка безубыточности в течении месяца достигается при наличии 800 активных подписчиков. В зависимости от количества пользователей скорость возврата инвестиций варьируется.

Для расчета доходов в месяц берется среднее количество подписчиков, 342 пользователей, учитывая, что подписка стоит 300 рублей, выходит:

]

Для расчета коэффициента экономической эффективности, необходимо учитывать экономическую эффективность и затраты на разработку приложения.

Для расчета срока окупаемости, необходимо учитывать доход в месяц и затраты на разработку приложения.

Подставив значения, получаем срок окупаемости.

При наличии 342 подписчиков ежемесячный доход составляет рублей, таким образом срок окупаемости составил месяцев. Таблица 4.3.1.

Чем выше число активных пользователей, тем быстрее компенсируются затраты на разработку и начинается получение чистой прибыли.

Таблица 4.3.1 – Показатели экономической эффективности

|  |  |
| --- | --- |
| Показатель | Величина |
| Трудозатраты на разработку проекта, чел\*час | 294 |
| Затраты на разработку проекта, руб. | 474726,75 |
| Экономическая эффективность проекта, руб. | 102 600 |
| Коэффициент экономической эффективности | 0,22 |
| Срок окупаемости, мес. | 4,6 |

Таким образом, произведенный экономический анализ эффективности создания и приложения доказывает целесообразность использования.

# Заключение

Современные технологии и цифровизация оказывают значительное влияние на образовательные процессы в различных областях знаний, и обучение графическому дизайну не является исключением. В условиях стремительного развития цифровых инструментов, роста востребованности специалистов в сфере графического дизайна и повышенных требований рынка труда, обучение данной дисциплине требует новых подходов и современных решений

В процессе разработки данного дипломного проекта было создано мобильное приложение для обучения графическому дизайну посредством изучения теорий и выполнение практических заданий. Были выполнены следующие задачи:

* Проведен анализ предметной области и проектирование информационной системы, направленной на обучение графическому дизайну;
* Построены диаграммы прецедентов, последовательности и «сущность-связь», обеспечивающие структурированное представление системы и компонентов;
* Подключена база данных Firebase для хранения информации о пользователях, прогрессе, выполненных заданиях и результатах обучения;
* Создан пользовательский интерфейс, адаптированный под образовательный процесс и удобный для различных категорий пользователей;
* Разработан программный код, реализующий основные функции системы.

Разработка интерфейса приложения также стала важной задачей, поскольку это должен быть интуитивно понятным и удобным для пользователя, обеспечивая простоту навигации и эффективное взаимодействие с функционалом образовательной платформы.

**Литература**

1. Рогов Е. В. Unity и C#: электронное издание / Бонд Джереми Гибсон – М.: Питер, 2022г.- 928 с.
2. Unity в действии/ Хокинг Джозеф.: электронное издание  
   – М.: Москва, 2023.- 448 с.
3. Корнилов А. В. Unity Полное руководство. Часть 1. / Наука и техника – М.: Москва, 2021.- 496 с.

Сайты сети «Интернет»

2. Unity Learn: электронно-библиотечная система: сайт, 2020-2025 -, - URL: https://learn.unity.com/ (дата обращения 29.04.2025). – Режим доступа: для всех пользователей. – Текст: электронный.

3. Первые шаги в Unity: электронно-библиотечная система: сайт, 2020-2025 -, - URL: https://habr.com/ru/companies/otus/articles/576434/ (дата обращения 19.04.2025). - Режим доступа: для всех пользователей. – Текст: электронный.

4. C# Learn: электронно-библиотечная система: сайт, 2020-2025 -, - URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения 14.04.2025). - Режим доступа: для всех пользователей. – Текст: электронный.

5. Unity 6.1 User Manual: электронно-библиотечная система: сайт, 2020-2025 -, - URL: <https://docs.unity3d.com/Manual/index.html?ref=toptrafficsites> (дата обращения 15.04.2025). - Режим доступа: для всех пользователей. – Текст: электронный.

6. Unity 6.1 User Manual: электронно-библиотечная система: сайт, 2020-2025 -, - URL: https://unityhub.ru/manual/UnityManual (дата обращения 20.04.2025). - Режим доступа: для всех пользователей. – Текст: электронный.

# Приложение А Руководство пользователя

Для начала работы с приложением необходимо пройти авторизацию. На экране авторизации ввести логин и пароль. При успешной авторизации откроется главная страница приложения. Авторизация и регистрация на рисунке А.1.

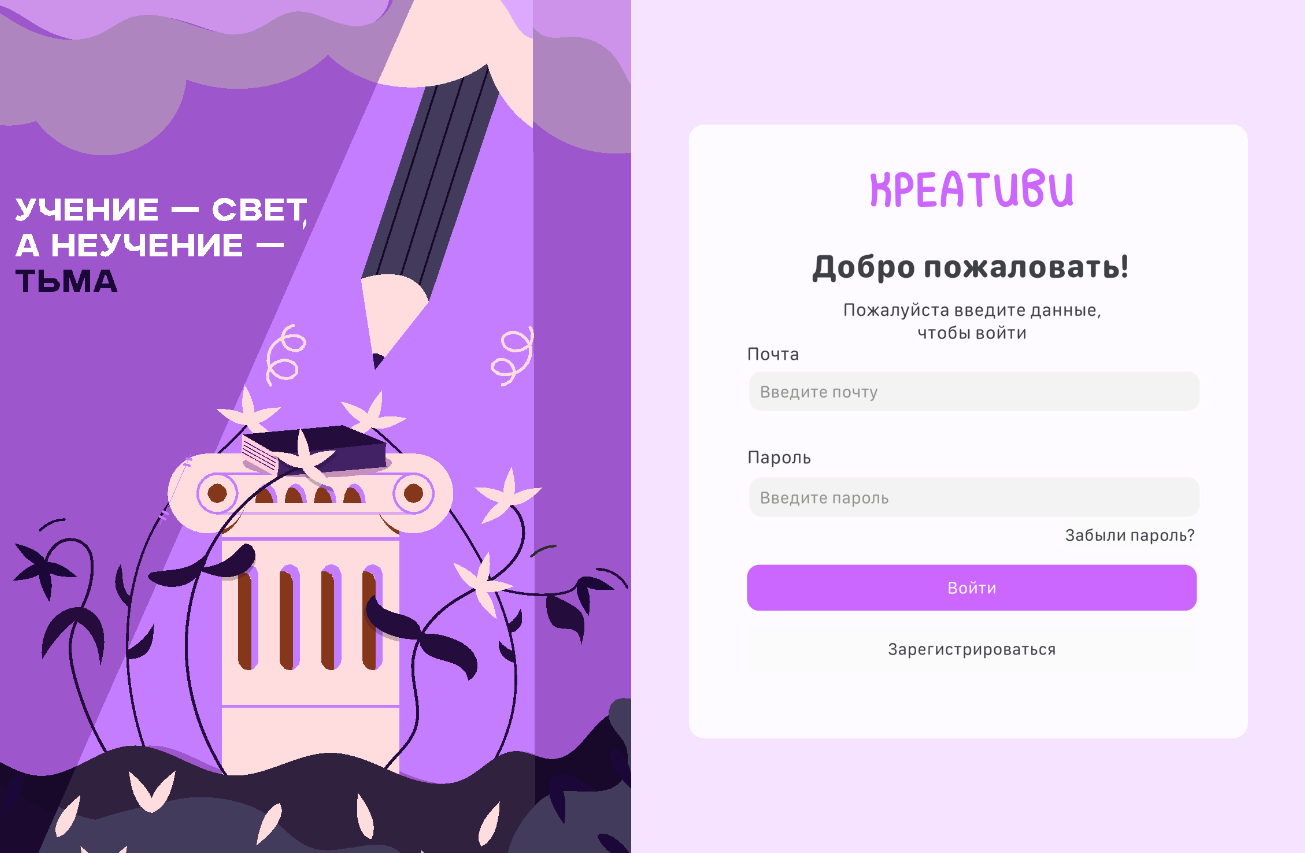


Рисунок А.1 – Авторизация

Главная страница представляет собой основной экран, на котором отображаются доступные курсы и задания. Здесь можно выбрать необходимый уровень обучения или перейти к завершению текущего задания. Главная страница на рисунке А.2.

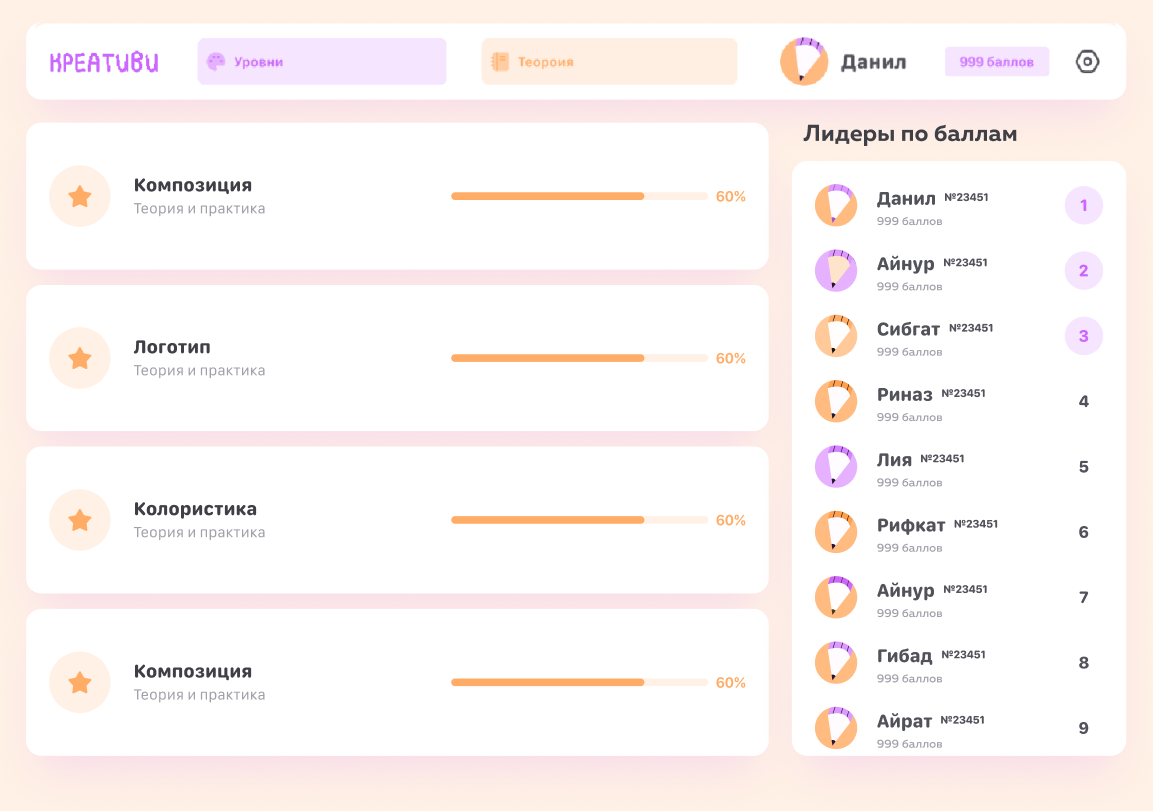


Рисунок А.2 – Главная страница

Прежде чем приступить к выполнению задания, появится окно с запросом на подтверждение. Окна на рисунке А.3.

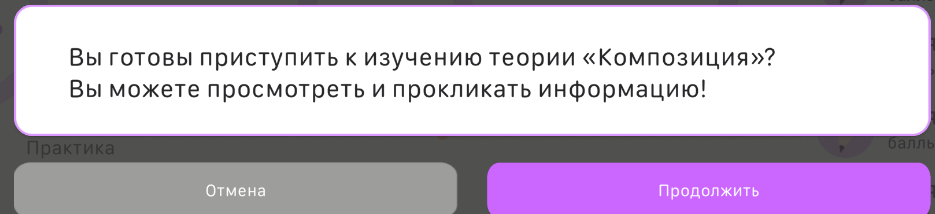


Рисунок А.3 – Подтверждение

Каждый уровень начинается с теории, которые помогут пользователю ознакомиться с практическими задачами. На этом этапе предоставляется информация о необходимых действиях для успешного прохождения уровня. Подтверждение на рисунке А.4.

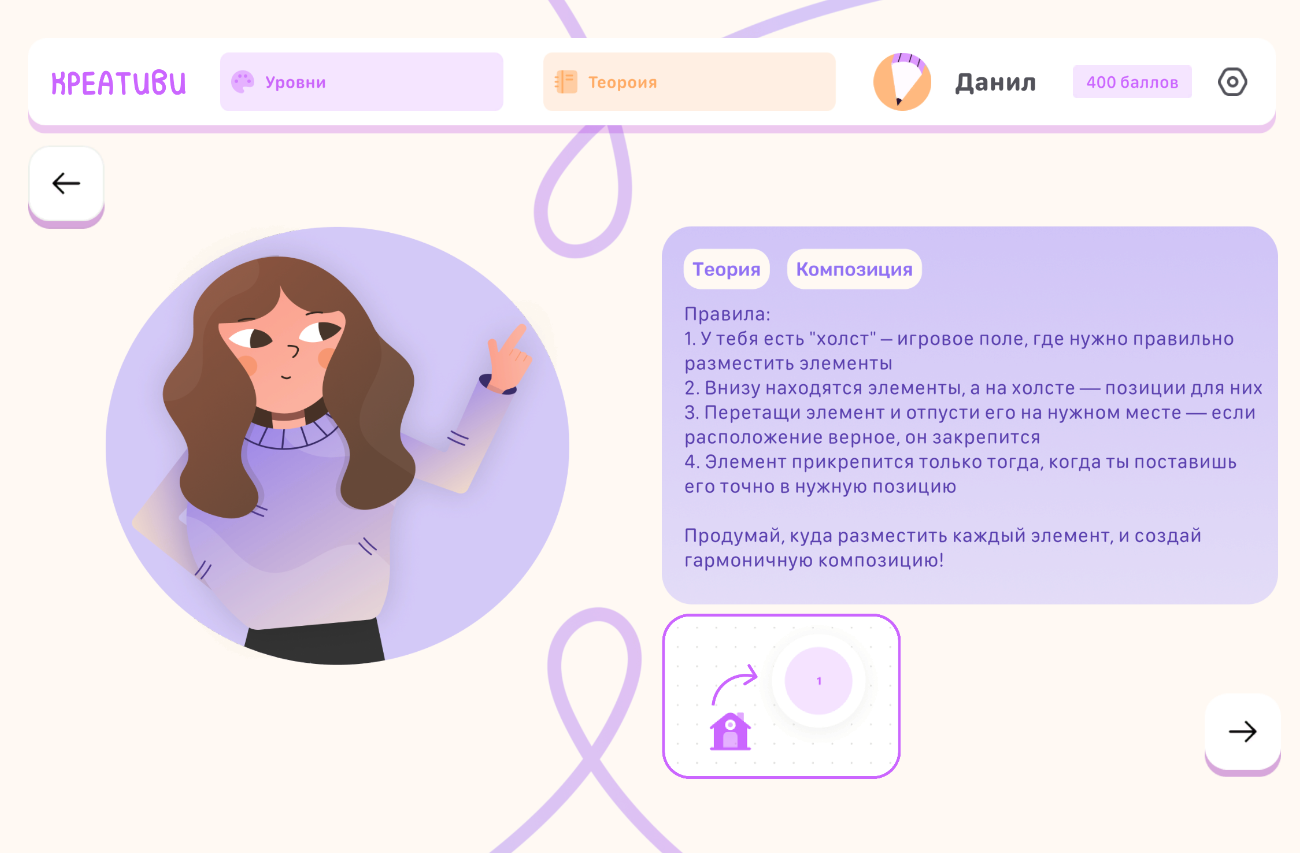


Рисунок А.4 – Страница с теорией

Уровень состоит из теоретических и практических заданий. На этом этапе пользователь должен выполнить серию заданий, чтобы продвигаться дальше. Страница теста и интерактивного задания на рисунке А.5-А.6.

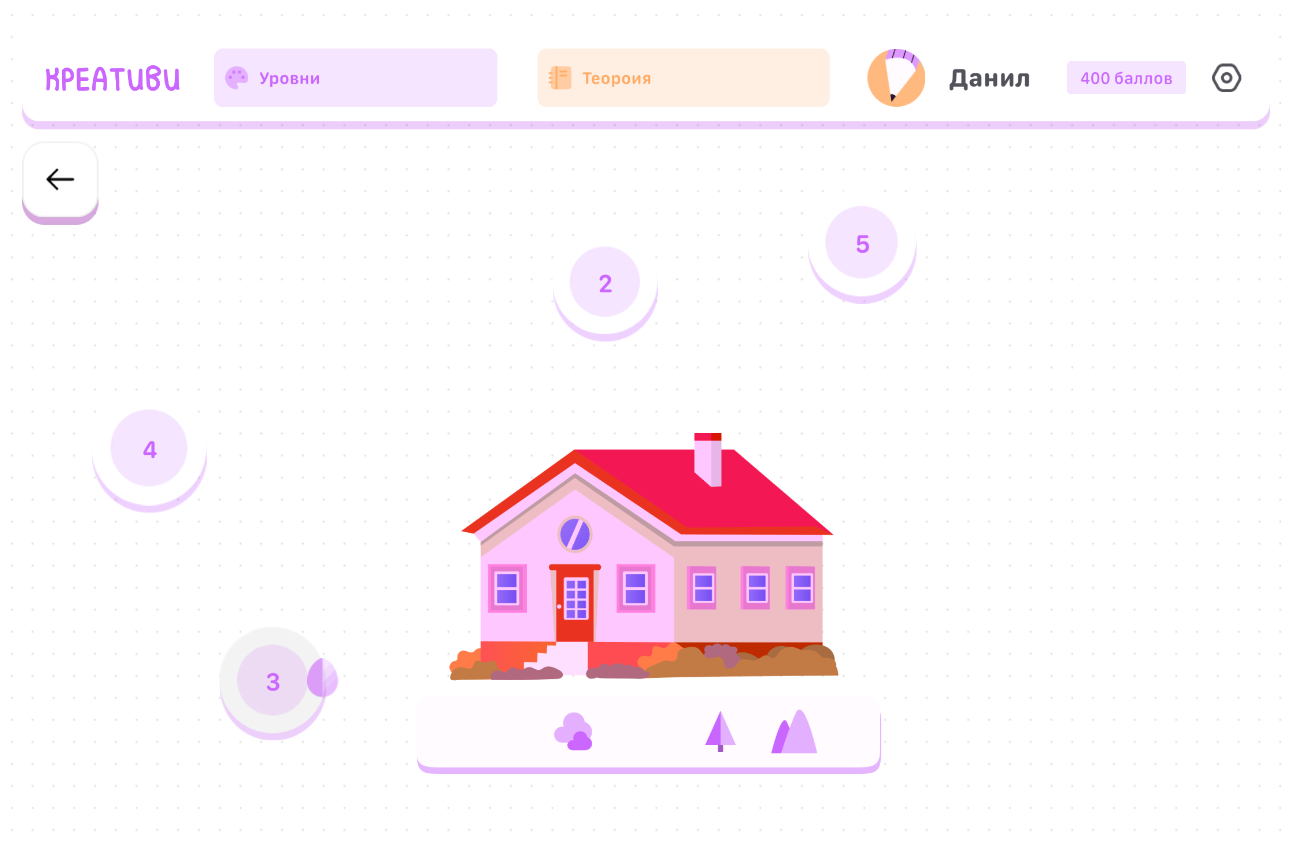


Рисунок А.5 – Интерактивный уровень

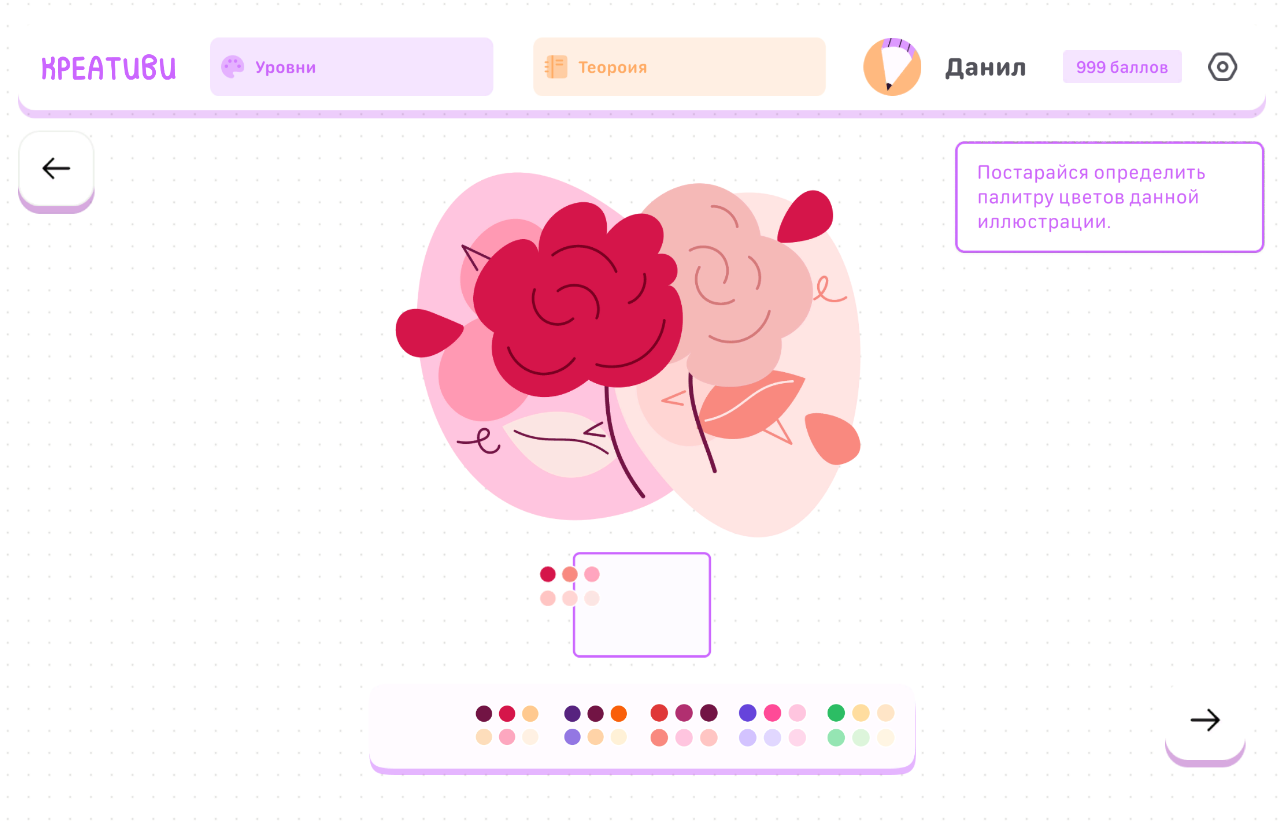


Рисунок А.6 – Интерактивный уровень

# Приложение Б Руководство программиста

В структуре проекта находятся четыре основные папки, каждая из которых выполняет определенную функцию.

Первая папка — kreativy. Содержит все исходные файлы, связанные с графическим контентом приложения. В этой папке находятся элементы, используемые в интерфейсе и дизайне, такие как изображения, иконки, фоны и другие графические ресурсы, которые могут быть использованы в процессе разработки приложения.

Вторая папка — backup. Здесь хранятся резервные копии всех ключевых файлов проекта. Папка предназначена для обеспечения безопасности данных и сохранности в случае необходимости восстановления утраченных файлов.

Третья папка — illustrator. В данной папке находятся файлы, созданные в Adobe Illustrator. Это основные файлы графики, которые используются для создания векторных элементов интерфейса, логотипов, а также других графических объектов, требующих точности и масштабируемости.

Четвертая папка — app. В этой папке хранится файл APK, предназначенный для установки на планшеты. Это сборка приложения, которая включает в себя все необходимые компоненты для работы на Android-устройствах. Папка содержит все данные, требуемые для запуска и функционирования приложения на конечных устройствах.

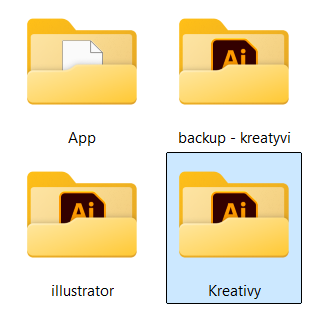


Рисунок Б.1 – Корневые папки

В проекте предусмотрены несколько скриптов, которые обеспечивают функциональность приложения, автоматизируют различные процессы и упрощают взаимодействие между компонентами.

Кроме того, существуют вспомогательные скрипты, которые отвечают за выполнение служебных задач, таких как настройка анимации, загрузка внешних данных или управление сетевыми соединениями. Скрипты на рисунке Б.2

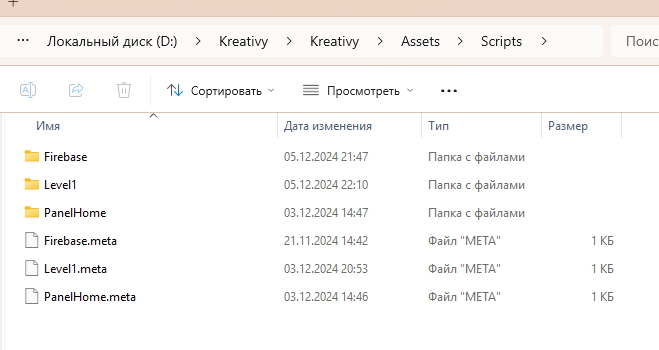


Рисунок Б.2 – Расположение скриптов

Файл kreativy является важной частью проекта и предназначен для экспорта в Unity. Файл содержит все графические ресурсы, которые будут использованы в приложении. В Unity этот файл будет импортирован в виде набора различных объектов, таких как изображения, текстуры и анимации. Эти ресурсы обеспечивают визуальную составляющую приложения и влияют на внешний вид и функциональность. Папка на рисунке Б.3.

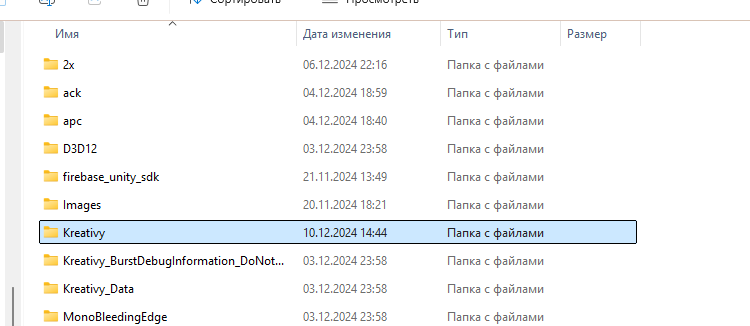


Рисунок Б.3 – Unity-файл

# Приложение B Программный код

Листинг 1 – Проверка и инициализация зависимостей Firebase, авторизация пользователя.

using UnityEngine;

using UnityEngine.UI;

using Firebase;

using Firebase.Auth;

using Firebase.Firestore;

using System.Threading.Tasks;

using TMPro;

using System.Collections;

using UnityEngine.SceneManagement;

using System.Collections.Generic;

public class FirebaseAuthManager : MonoBehaviour

{

    [Header("UI поля")]

    public TMP\_InputField emailInputField;

    public TMP\_InputField passwordInputField;

    public Button loginButton;

    [Header("Тексты ошибок")]

    public TMP\_Text emailErrorLabel;

    public TMP\_Text passwordErrorLabel;

    [Header("Иконки ошибок")]

    public GameObject emailErrorIcon;

    public GameObject passwordErrorIcon;

    [Header("Спрайты для инпутов")]

    public Sprite defaultInputSprite;

    public Sprite errorInputSprite;

    private FirebaseAuth auth;

    private FirebaseFirestore db;

    private FirebaseUser user;

    private bool isFirebaseInitialized = false;

    private Coroutine initCoroutine;

    void Start()

    {

        loginButton.onClick.AddListener(LoginButtonClick);

        SetUIInteractable(false);

        HideErrorUI();

        initCoroutine = StartCoroutine(InitializeFirebaseWithTimeout());

    }

    IEnumerator InitializeFirebaseWithTimeout()

    {

        var dependencyTask = FirebaseApp.CheckAndFixDependenciesAsync();

        yield return new WaitUntil(() => dependencyTask.IsCompleted);

        if (dependencyTask.Exception != null)

        {

            Debug.LogError("Ошибка при проверке зависимостей Firebase");

            yield break;

        }

        var initializationTask = InitializeFirebaseAsync();

        float timeout = 10f;

        float elapsedTime = 0f;

        while (!initializationTask.IsCompleted && elapsedTime < timeout)

        {

            elapsedTime += Time.deltaTime;

            yield return null;

        }

        if (!isFirebaseInitialized)

        {

            Debug.LogError("Firebase не был инициализирован в установленный срок");

            yield break;

        }

        SetUIInteractable(true);

    }

    async Task InitializeFirebaseAsync()

    {

        try

        {

            auth = FirebaseAuth.DefaultInstance;

            auth.StateChanged += AuthStateChanged;

            db = FirebaseFirestore.DefaultInstance;

            await Task.Delay(500);

            isFirebaseInitialized = true;

        }

        catch (System.Exception ex)

        {

            Debug.LogError($"[Ошибка] Инициализация Firebase: {ex}");

        }

    }

    public async void LoginButtonClick()

    {

        if (!ValidateInputs()) return;

        SetUIInteractable(false);

        try

        {

            string email = emailInputField.text;

            string password = passwordInputField.text;

            var loginTask = auth.SignInWithEmailAndPasswordAsync(email, password);

            await loginTask;

            if (loginTask.IsCompletedSuccessfully)

            {

                user = auth.CurrentUser;

                await LoadUserData();

                SceneManager.LoadScene("Home");

            }

            else

            {

                HandleFirebaseError(loginTask.Exception);

            }

        }

        catch (FirebaseException ex)

        {

            Debug.LogError($"[Ошибка Firebase] {ex.Message}");

            HandleFirebaseError(ex);

        }

        finally

        {

            SetUIInteractable(true);

        }

    }

    private async Task LoadUserData()

    {

        if (user == null)

        {

            Debug.LogError("user == null");

            return;

        }

        DocumentReference docRef = db.Collection("users").Document(user.UserId);

        DocumentSnapshot snapshot = await docRef.GetSnapshotAsync();

        if (snapshot.Exists)

        {

            Dictionary<string, object> userData = snapshot.ToDictionary();

            if (userData.ContainsKey("Name") && userData.ContainsKey("email") && userData.ContainsKey("Balls"))

            {

                string name = userData["Name"]?.ToString();

                string email = userData["email"]?.ToString();

                long balls = userData.ContainsKey("Balls") ? (long)userData["Balls"] : 0;

                UserData.Instance.SetUserData(name, email, balls);

            }

        }

    }

    void HandleFirebaseError(System.Exception exception)

    {

        FirebaseException firebaseEx = exception as FirebaseException;

        if (firebaseEx != null)

        {

            AuthError errorCode = (AuthError)firebaseEx.ErrorCode;

            switch (errorCode)

            {

                case AuthError.InvalidEmail:

                    ShowEmailError("Неверный email");

                    break;

                case AuthError.WrongPassword:

                    ShowPasswordError("Неверный пароль");

                    break;

                case AuthError.UserNotFound:

                    ShowEmailError("Пользователь не найден");

                    break;

                default:

                    ShowEmailError("Неверный email");

                    ShowPasswordError("Неверный пароль");

                    break;

            }

        }

    }

    bool ValidateInputs()

    {

        bool isValid = true;

        string email = emailInputField.text.Trim();

        string password = passwordInputField.text;

        if (string.IsNullOrWhiteSpace(email))

        {

            ShowEmailError("Введите email");

            isValid = false;

        }

        else if (!IsValidEmail(email))

        {

            ShowEmailError("Неверный формат email");

            isValid = false;

        }

        else

        {

            HideEmailError();

        }

        if (string.IsNullOrWhiteSpace(password))

        {

            ShowPasswordError("Введите пароль");

            isValid = false;

        }

        else

        {

            HidePasswordError();

        }

        return isValid;

    }

    void ShowEmailError(string message)

    {

        emailErrorLabel.text = message;

        emailErrorLabel.gameObject.SetActive(true);

        emailErrorIcon.SetActive(true);

        SetInputSprite(emailInputField, errorInputSprite);

    }

    void ShowPasswordError(string message)

    {

        passwordErrorLabel.text = message;

        passwordErrorLabel.gameObject.SetActive(true);

        passwordErrorIcon.SetActive(true);

        SetInputSprite(passwordInputField, errorInputSprite);

    }

    void HideEmailError()

    {

        emailErrorLabel.gameObject.SetActive(false);

        emailErrorIcon.SetActive(false);

        SetInputSprite(emailInputField, defaultInputSprite);

    }

    void HidePasswordError()

    {

        passwordErrorLabel.gameObject.SetActive(false);

        passwordErrorIcon.SetActive(false);

        SetInputSprite(passwordInputField, defaultInputSprite);

    }

    void HideErrorUI()

    {

        HideEmailError();

        HidePasswordError();

    }

    void SetInputSprite(TMP\_InputField inputField, Sprite sprite)

    {

        Image image = inputField.GetComponent<Image>();

        if (image != null)

        {

            image.sprite = sprite;

        }

    }

    bool IsValidEmail(string email)

    {

        try

        {

            var addr = new System.Net.Mail.MailAddress(email);

            return addr.Address == email;

        }

        catch

        {

            return false;

        }

    }

    void SetUIInteractable(bool interactable)

    {

        emailInputField.interactable = interactable;

        passwordInputField.interactable = interactable;

        loginButton.interactable = interactable;

    }

    void OnDestroy()

    {

        if (auth != null)

        {

            auth.StateChanged -= AuthStateChanged;

            auth = null;

        }

        if (initCoroutine != null)

        {

            StopCoroutine(initCoroutine);

        }

    }

    void AuthStateChanged(object sender, System.EventArgs eventArgs)

    {

        if (!isFirebaseInitialized) return;

        if (auth.CurrentUser != user)

        {

            user = auth.CurrentUser;

        }

    }

}

Листинг 2 – Обработка нажатия кнопки входа

public async void LoginButtonClick()

{

if (!ValidateInputs()) return;

SetUIInteractable(false);

try

{

string email = emailInputField.text;

string password = passwordInputField.text;

var loginTask = auth.SignInWithEmailAndPasswordAsync(email, password);

await loginTask;

if (loginTask.IsCompletedSuccessfully)

{

user = auth.CurrentUser;

await LoadUserData();

SceneManager.LoadScene("Home");

}

else

{

HandleFirebaseError(loginTask.Exception);

}

}

catch (FirebaseException ex)

{

Debug.LogError($"[Ошибка Firebase] {ex.Message}");

HandleFirebaseError(ex);

}

finally

{

SetUIInteractable(true);

}

}

Листинг 3 – Загрузка пользовательских данных из Firestore

bool ValidateInputs()

{

bool isValid = true;

string email = emailInputField.text.Trim();

string password = passwordInputField.text;

if (string.IsNullOrWhiteSpace(email))

{

ShowEmailError("Введите email");

isValid = false;

}

else if (!IsValidEmail(email))

{

ShowEmailError("Неверный формат email");

isValid = false;

}

else

{

HideEmailError();

}

if (string.IsNullOrWhiteSpace(password))

{

ShowPasswordError("Введите пароль");

isValid = false;

}

else

{

HidePasswordError();

}

return isValid;

}

Листинг 4 – Обработка ошибок аутентификации Firebase

void HandleFirebaseError(System.Exception exception)

{

FirebaseException firebaseEx = exception as FirebaseException;

if (firebaseEx != null)

{

AuthError errorCode = (AuthError)firebaseEx.ErrorCode;

switch (errorCode)

}

}}