

**МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Факультет компьютерных наук

Техническое задание в соответствии с ГОСТ 34.602-89

Исполнители

\_\_\_\_\_ Богачев-Воевудский А.

\_\_\_\_\_ Веремеев В.

\_\_\_\_\_ Габелко М.

\_\_\_\_\_ Штукатуров Д.

\_\_\_\_\_ Елфимов А.

\_\_\_\_\_ Дубровин Д.

Заказчик

\_\_\_\_\_ Тарасов В.С.

Воронеж 2025

# СОДЕРЖАНИЕ

1 Предпроектный исследование .....	5
1.1 Анализ рынка.....	5
1.2 Целевая аудитория .....	5
1.3 Основные конкуренты.....	5
2 Техническое задание (ТЗ) .....	6
2.1 Цели и задачи проекта.....	6
2.1.1 Цели.....	6
2.1.2 Задачи.....	6
2.1.3 Критерии успешности .....	6
2.2 Функциональные и нефункциональные требования .....	6
2.2.1 Функциональные требования.....	6
2.2.2 Нефункциональные требования.....	8
2.3 Пользовательские сценарии (User Stories) .....	8
2.3.1 Начало игры .....	8
2.3.2 Ловля рыбы .....	9
2.3.3 Улучшение удочки .....	9
2.3.4 Бои с боссами.....	10
2.4 Перечень основных функциональных блоков системы .....	10
2.4.1 Система управления игроком .....	10
2.4.2 Система рыбалки .....	10
2.4.3 Система улучшений.....	10
2.4.4 Система магазина .....	11
2.4.5 Система боссов .....	11
2.4.6 Система сохранения .....	11
2.4.7 Система коллекционирования .....	11
3 Начальная архитектура .....	12
3.1 UML диаграммы .....	12
3.2 ER диаграмма.....	13
3.3 Схема API .....	14
3.3.1 Описание основных энд поинтов.....	14

3.3.2 Взаимодействие компонентов .....	15
3.4 Предварительный выбор стека технологий .....	16
3.4.1 Backend.....	16
3.4.2 Frontend (Игра):.....	16
3.4.3 Хранение данных:.....	17
4 Дизайн макеты.....	18
4.1 UI Kit .....	18
4.1.1 Цветовая палитра:.....	18
4.1.2 Цветовые акценты: .....	18
4.1.3 Шрифты: .....	18
4.2 Брендбук .....	20
4.2.1 Логотип: .....	20
4.2.2 Фирменный стиль:.....	20
4.2.3 Анимации:.....	21
4.2.4 Звуки:.....	21
4.2.5 Дополнительные элементы: .....	21
4.2.6 Элементы декора: .....	21
5 Организация проекта.....	22
5.1 Настройка гита.....	22
5.1.1 Гит репозиторий .....	22
6 Ограничения проекта и технические риски .....	23
6.1 Ограничения проекта .....	23
6.1.1 Этап 1: Проектирование архитектуры, написание документации, анализ требований. (1-2 недели).....	23
6.1.2 Этап 2: Разработка MVP (3-4 недели) .....	24
6.1.3 Этап 3: «Полировка» MVP и устранение проблем (3-4 недели)...	25
6.2 Технические риски .....	25
6.2.1 Риски с стороны Frontend.....	25
6.2.2 Риски с стороны Backend .....	26

## Термины и определения

**Десктоп** – компьютер.

**Пиксель-арт** – графика, стилизованная под старые игры, где графика состояла из крупных пикселей.

**RPG (Role Play Game)** – тип игр, в котором игрок отыгрывает определенного персонажа и ему предоставлен выбор действий.

**FPS (Frame Per Second)** – количество кадров в секунду, наглядно отображающее производительность на устройстве.

**Юзабилити** – удобство / пригодность использования приложения.

**Логирование** — это фиксация событий в работе приложения, помогающая его разработчикам выявлять баги системы.

**User stories** – пользовательские сценарии, используемые для планирования сессии пользователя приложения.

**Acceptance Criteria** – критерии приемки, важная практика для улучшения коммуникации между разработчиками и заказчиками, а также неотъемлемая часть создания качественных пользовательских сценариев.

**UI Kit** - набор готовых решений пользовательского интерфейса.

**Иконографика** — это графический способ подачи информации, данных и знаний, целью которого является быстро и чётко преподнести сложную информацию.

# **1 Предпроектный исследование**

## **1.1 Анализ рынка**

- Рынок мобильных и десктопных игр в жанре "коллекционирование" активно развивается.
- Пиксель-арт уменьшает время на создание объектов и улучшает производительность.
- Игры с элементами RPG (улучшение инвентаря, открытие новых локаций) пользуются популярностью.

## **1.2 Целевая аудитория**

- Возраст: 12+ лет.
- Интересы: коллекционирование, рыбалка, пиксель-арт, RPG-элементы.

## **1.3 Основные конкуренты**

- "Stardew Valley" (хотя это фермерская игра, но содержит механику рыбной ловли и схожая рисовка).
- "Fishing Planet" (серьезная симуляция, но без пиксель-арта).
- "Fishdom" (фокус на декорировании аквариума).

## **2 Техническое задание (ТЗ)**

### **2.1 Цели и задачи проекта**

#### **2.1.1 Цели**

- Создать увлекательную игру в жанре "рыбалка", сочетающую коллекционирование, исследование и улучшение инвентаря.
- Реализовать пиксель-арт стиль для уникального визуального опыта.
- Привлечь аудиторию для повышения доходов от монетизации

#### **2.1.2 Задачи**

- Разработать геймплей и интерфейс игры
- Создать базу данных для хранения прогресса игрока.
- Реализовать механизм генерации истории-лора

#### **2.1.3 Критерии успешности**

- Удержание игроков на 30 минут – 1 час в день.
- Позитивные отзывы о графике и игровом процессе.
- Монетизация через рекламу или внутриигровые микротранзакции.

### **2.2 Функциональные и нефункциональные требования**

#### **2.2.1 Функциональные требования**

- **Сохранение прогресса:** при повторном заходе игрока нужно дать ему возможность продолжить игру там, где он остановился. Прогресс игрока будет сохраняться через создание аккаунта внутри игры и внесением этой информации в базу данных.
- **Алгоритм изменения сложности:** в процессе игры от нескольких факторов будет зависеть сложность игрового процесса.
- **Возможность начать ловлю рыбы с любой части водоема:** игроку не нужно искать одну точку взаимодействия на весь водоем.
- **Выбор наживки:** можно будет купить наживку для ловли рыбы.
- **Различные характеристики у рыб:** вероятность поймать рыбу от наживки, времени суток и прокачки удочки.
- **Улучшение удочки:** игрок может исследовать мир и находить улучшения для удочки, влияющие на баланс.
- **Продажа рыбы:** можно будет продать рыбу для покупки наживки или улучшения удочки.
- **Исследование мира:** для исследования мира есть сундуки, разбросанные по локации и содержащие в себе улучшения, которые нельзя купить в магазине.
- **Инвентарь:** игрок сам выбирает какую рыбу он хочет продать.
- **Достижения:** за выполнение определенных условий, не всегда видимых игроку, выдается достижение.
- **Тайловая система:** Карта реализуется как 2D-сетка тайлов с координатами (x,y). Каждый тайл содержит данные о типе terrain и интерактивных объектах.

- **Механика взаимодействия:** Взаимодействие с объектами происходит через проверку соседних тайлов в направлении взгляда игрока. Радиус взаимодействия: 1 тайл от текущей позиции.
- **Генерация истории мира на записках:** по локации будут расставлены сундуки с улучшениями для удочки и иногда в них будут лежать записки, текст на который генерируется при старте новой игры.

### 2.2.2 Нефункциональные требования

- **Масштабируемость:** возможность добавить новые локации и предметы.
- **Производительность:** должно быть не менее 30 FPS в игре и возможность включить 60 FPS на мобильных телефонах с ОС Android.
- **Юзабилити:** игроку должно быть интуитивно понятно, как работают механики в игре.
- **Локализация:** возможность игры на русском и английском языке.
- **Логирование данных:** нужно логировать данные для сбора статистики об игроках.

## 2.3 Пользовательские сценарии (User Stories)

### 2.3.1 Начало игры

- **Как:** Новый игрок.
- **Что хочу:** начать играть.



- **Почему:** чтобы начать процесс коллекционирования.
- **Acceptance Criteria:**
  - Игрок попадает на стартовый экран.

### 2.3.2 Ловля рыбы

- **Как:** Игрок.
- **Что хочу:** ловить рыбу с использованием удочки.
- **Почему:** чтобы собирать коллекцию и получать деньги.
- **Acceptance Criteria:**
  - Игрок может выбирать место для рыбалки.
  - Процесс ловли включает мини-игру.
  - Игрок получает рыбу после успешной ловли.

### 2.3.3 Улучшение удочки

- **Как:** Игрок.
- **Что хочу:** найти улучшения для удочки.
- **Почему:** чтобы сделать процесс ловли проще.
- **Acceptance Criteria:**
  - Игрок может исследовать карту.
  - На карте есть точки с улучшениями.
  - Игрок может применить найденные улучшения.

### **2.3.4 Бои с боссами**

- **Как:** Игрок.
- **Что хочу:** победить рыбу-босса.
- **Почему:** чтобы открыть новую локацию.
- **Acceptance Criteria:**
  - Игрок может вызвать босса после выполнения условий.
  - Бой включает специальные механики.
  - После победы открывается новая локация.

## **2.4 Перечень основных функциональных блоков системы**

### **2.4.1 Система управления игроком**

- Перемещение по локации
- Взаимодействие с интерактивными объектами (водоем, магазин, сундуки)

### **2.4.2 Система рыбалки**

- Мини-игра при взаимодействии с водоемом
- Паттерн поведения у рыб

### **2.4.3 Система улучшений**

- Поиск и применение улучшений для удочки.

#### **2.4.4 Система магазина**

- Покупка наживки
- Продажа рыбы
- Выдача заданий игроку

#### **2.4.5 Система боссов**

- Бой с боссами в виде усложненной мини-игры
- Открытие новых локаций
- Мини-игра перед боем

#### **2.4.6 Система сохранения**

- Сохранение прогресса игрока.
- Возможность стереть сохранение
- Загрузка прогресса при входе

#### **2.4.7 Система коллекционирования**

- Журнал пойманной рыбы

- Отображение в журнале собранной коллекции рыб

### 3 Начальная архитектура

#### 3.1 UML диаграммы

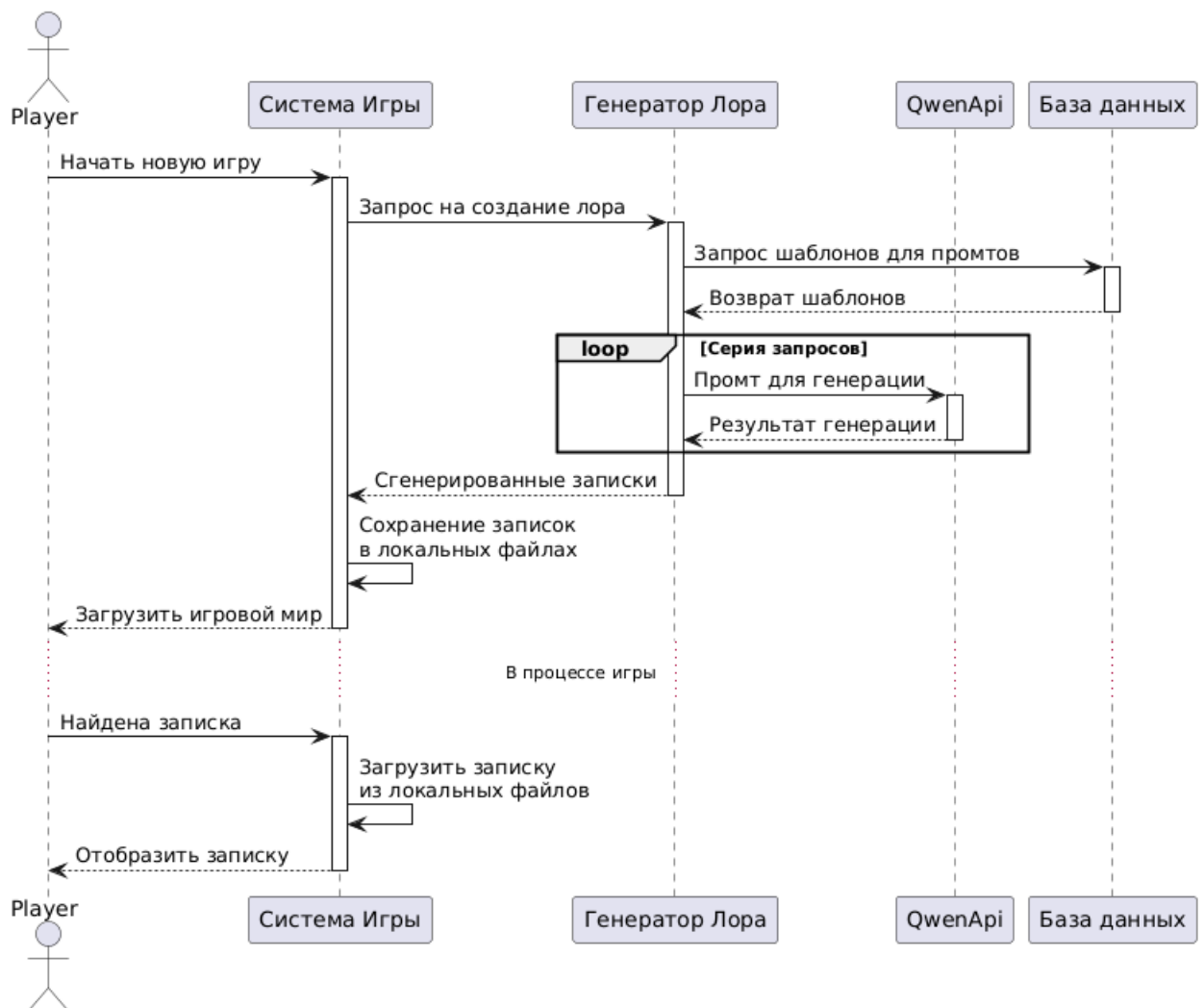


Рисунок 1 Диаграмма последовательностей

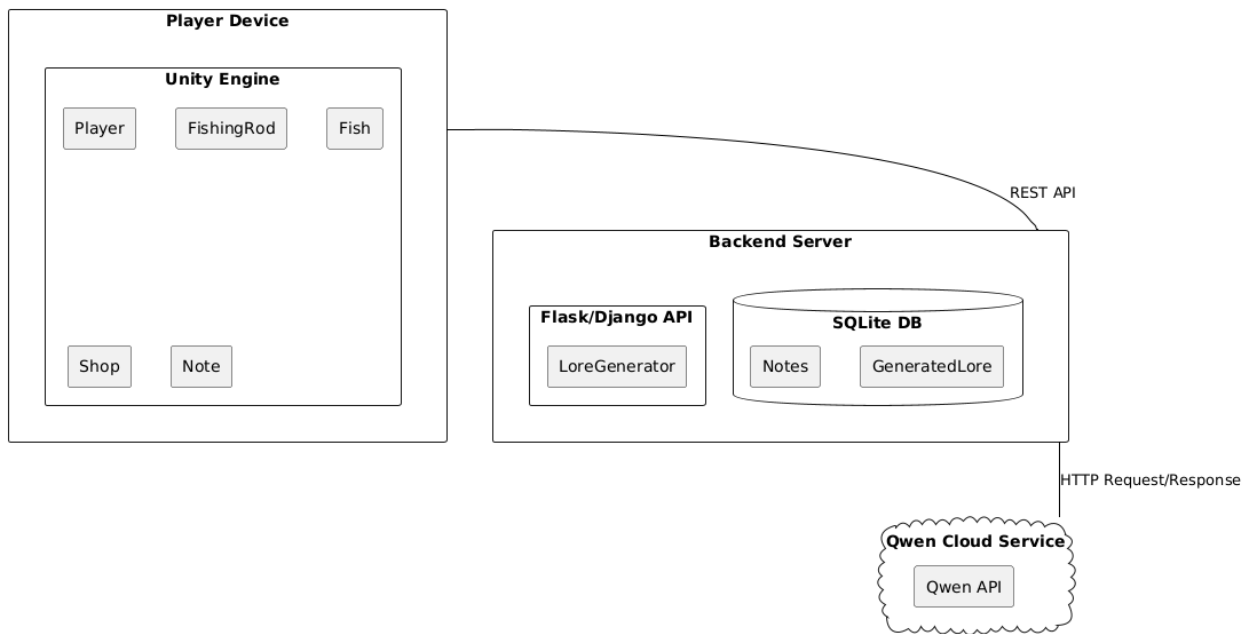


Рисунок 2 Диаграмма развёртывания

## 3.2 ER диаграмма

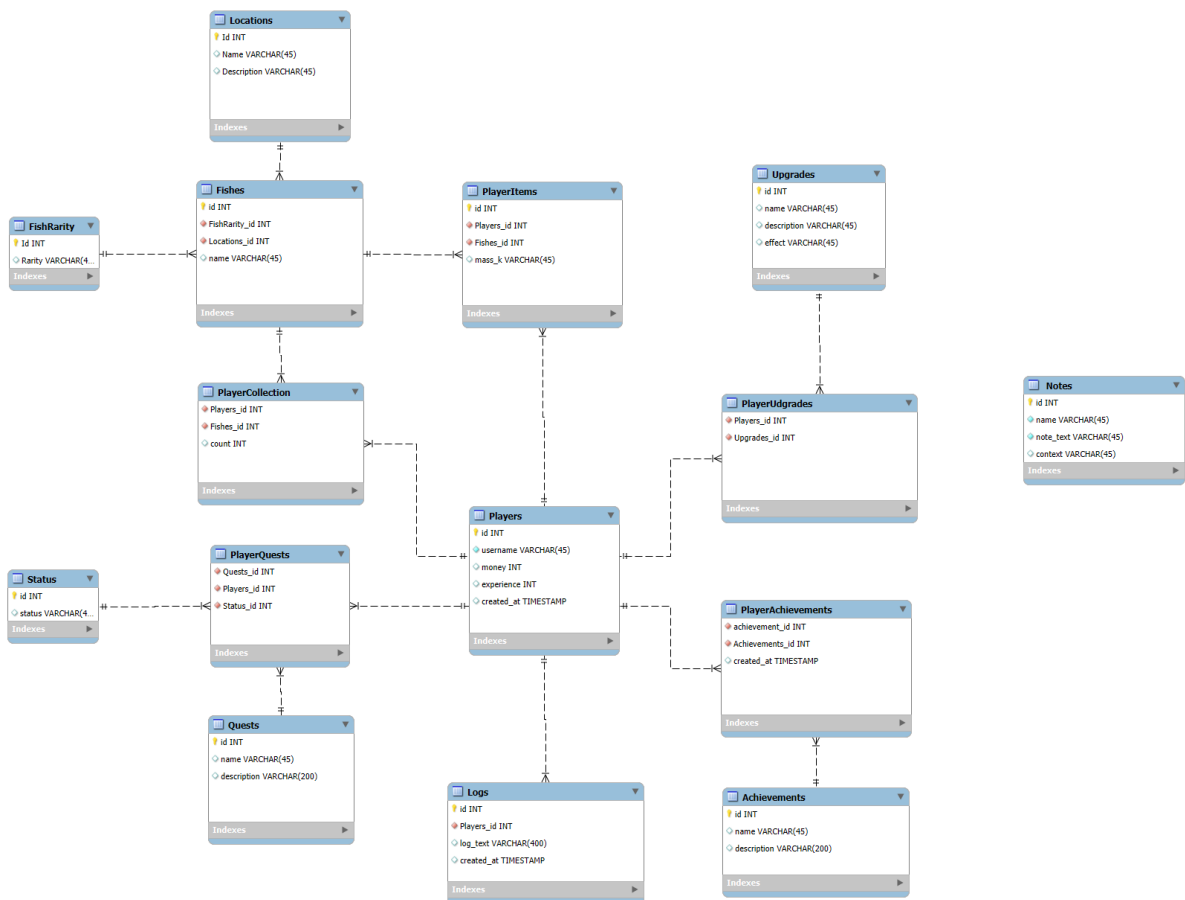


Рисунок 3 ER диаграмма БД

### 3.3 Схема API

#### 3.3.1 Описание основных энд поинтов

##### 1. Аутентификация и пользователи

Эта секция содержит методы для управления учетными записями пользователей, включая регистрацию, авторизацию и обновление токенов.

##### Эндпоинты:

- POST /api/auth/register
  - Описание: Создание нового аккаунта пользователя с указанием логина, пароля и языка интерфейса. Требуемые параметры: **login, password, language**.
  - Ответ: Статус регистрации и уникальный идентификатор пользователя (**user\_id**).
- POST /api/auth/login
  - Описание: Авторизация существующего пользователя. Возвращает данные для входа (например, токен). Требуемые параметры: **login, hash\_password**.
  - Ответ: Токен доступа (**access\_token**) и его срок действия (**expires\_in**).
- POST /api/auth/refresh
  - Описание: Обновление истекшего токена доступа. Требуемые параметры: **refresh\_token**.
  - Ответ: Новый токен доступа (**access\_token**) и его срок действия (**expires\_in**).

##### 2. Достижения

Эта секция предоставляет методы для работы с достижениями пользователя, включая просмотр разблокированных достижений и их активацию.

#### **Эндпоинты:**

- GET /api/achievements
  - Описание: Получение списка достижений, которые были разблокированы пользователем.  
Требуемые параметры: user\_id (передается через заголовок Authorization).
  - Ответ: Массив достижений с их названием, описанием и датой разблокировки.
- POST /api/achievements/unlock
  - Описание: Разблокировка достижения при выполнении определенного условия (например, поймано 10 рыб).  
Требуемые параметры: achievement\_id (ID достижения), user\_id.
  - Ответ: Подтверждение разблокировки (success: true) или сообщение об ошибке.

### **3.3.2 Взаимодействие компонентов**

#### **1. Клиент (Unity):**

- Отправляет запросы на бэкенд через REST API.
- Обрабатывает ответы (обновление UI, анимации).
- Локально кэширует данные для оффлайн-доступа (с последующей синхронизацией).

#### **2. Бэкенд:**

- Игровой сервер (Python):
  - Обработывает логику перемещения, взаимодействия, торговли.
  - Интегрируется с базой данных.
- Сервис генерации лора (отдельный микросервис):
  - Использует Qwen для создания уникальных записок.
- Аналитика:
  - Логирует действия игроков (ловля рыбы, покупки).

### 3. Базы данных:

- MySQL
  - Таблица Players.

## 3.4 Предварительный выбор стека технологий

### 3.4.1 Backend

- Язык: Python (FastAPI)
- База данных: MySQL.
- Аутентификация: по логину и хэшу пароля
- LLM-интеграция: Микросервис на Python (использует Qwen через API).

### 3.4.2 Frontend (Игра):

- Движок: Unity (C#).
- Локализация: Unity Localization Package.



### **3.4.3 Хранение данных:**

- Локальные: локальные файлы (с JSON)
- Серверные: MySQL

## **4 Дизайн макеты**

### **4.1 UI Kit**

#### **4.1.1 Цветовая палитра:**

- Основные цвета: Синий, зеленый, коричневый, оранжевый, красный, белый.
- Дополнительные цвета: Фиолетовый, темно-синий, ярко-желтый.

#### **4.1.2 Цветовые акценты:**

- Использование градиентов
- Контрастные цвета.

#### **4.1.3 Шрифты:**

- Pixelated шрифты
- Компоненты:
- Кнопки:
  - Основные кнопки: прямоугольная форма с закругленными углами, фон — градиент между коричневым и оранжевым, текст белый.
  - Второстепенные кнопки: более светлые тона (бежевый или светло-зеленый), текст черный.
  - Активные кнопки: добавление эффекта свечения (ярко-желтый контур).

- Меню:
  - Главное меню: фоновое изображение озера с плавающими рыбами, кнопки расположены вертикально по центру.
  - Настройки: слайдеры для регулировки громкости, переключатели для выбора режимов игры.



Рисунок 4 меню игры



Рисунок 5 Спрайт дома и персонажа

## 4.2 Брендбук

### 4.2.1 Логотип:

- Описание логотипа: Изображение кота в пиксель-арт стиле, держащего рыбу в лапах.
- Кот выполнен в розовых, рыба — в синих.

### 4.2.2 Фирменный стиль:

- Общий стиль:
  - Фэнтези-ретро в пиксельном стиле с акцентом на природу и рыбалку.
  - Элементы дизайна вдохновлены старыми 8-битными играми, но с современной адаптацией для удобства.
- Иконографика:
  - Все иконки выполнены в пиксель-арт стиле: удочка, рыбы, монеты.

- Иконки имеют четкие границы и минимальное количество деталей для сохранения читаемости.

#### **4.2.3 Анимации:**

- Легкие анимации для кнопок.
- Анимации передвижения персонажа.

#### **4.2.4 Звуки:**

- 8 ми-битные мелодии для фоновой музыки.
- Звуковые эффекты: плеск воды, кот мяукает при взаимодействии, звуки открывания дверей.

#### **4.2.5 Дополнительные элементы:**

- Пиксельные текстуры для создания объема и наложения теней.

#### **4.2.6 Элементы декора:**

- Пиксельные деревья, горы, камни, кусты, лед, трава, дома, предметы интерьера для создания атмосферы.
- Декоративные рамки для окон и меню, выполненные в стиле деревянных конструкций.

## **5 Организация проекта**

### **5.1 Настройка гита**

#### **5.1.1 Гит репозиторий**

Ссылка: [https://github.com/Aleygv/Fishing\\_game](https://github.com/Aleygv/Fishing_game)

## **6 Ограничения проекта и технические риски**

### **6.1 Ограничения проекта**

#### **6.1.1 Этап 1: Проектирование архитектуры, написание документации, анализ требований. (1-2 недели)**

**Цель:** расписать core-механики игры, определиться с визуальной частью игры и организовать архитектуру.

- Определение механик в игре (разработчик + аналитик)
  - Разбор механики перемещения персонажа
  - Разбор механики взаимодействия с интерактивными объектами (водоем, магазин, сундук)
  - Определение баланса игры (как исследование мира влияет на игру)
  - Разбор механики ловли рыбы
- Составление ТЗ (вся команда)
  - Составление функциональных и нефункциональных требований
  - Проектирование базы данных (разработчик БД)
  - Определение структуры хранения данных.
  - Интеграция с заранее продуманными запросами к Qwen для составления лора игры.
  - Выбор визуального стиля игры (художник + вся команда)
  - Определить визуальный стиль игры (пиксель-арт и т.п.).
  - Выбрать в каком пространстве делать (2D, 2.5D, 3D).
  - Определить UI в меню и в игре.

### 6.1.2 Этап 2: Разработка MVP (3-4 недели)

**Цель:** создать прототип игры с реализованными core-механиками.

- Разработка механики ловли рыбы (разработчики)
  - Разработать прототип ловли рыбы.
  - Разработать алгоритм увеличения сложности.
- Разработка механики инвентаря (разработчики)
  - Сделать возможность добавления / удаления объектов.
- Визуальный стиль (художник)
  - Отрисовать персонажа.
  - Анимировать персонажа.
  - Нарисовать ассеты для первой локации: дом, водоемы, сундуки, тропинки, деревья.
  - Расставить объекты на уровне.
- Описание core-механик (аналитик + разработчики)
  - Описание принципов механик
  - Оформление алгоритмов в Confluence
- Разработка backend составляющей
  - Создание базы данных и её настройка
  - Создать взаимодействие с API Qwen



### **6.1.3 Этап 3: «Полировка» MPV и устранение проблем (3-4 недели)**

**Цель:** добавить в игру больше графики и улучшить ощущения от взаимодействия с UI элементами.

- Улучшение визуальной составляющей (художники)
  - Добавить различные вариации рыб
  - Доработать окружение (перерисовать спрайты)
  - Добавить анимации в игру и элементам UI
- Связать frontend и backend (разработчики)
  - Настроить сохранение результатов в базу данных
  - Добавить взаимодействие с API Qwen в игру
  - Добавить аудио составляющую (вся команда)
  - Добавить звуки к UI элементам
  - Добавить звуки в игре'

## **6.2 Технические риски**

### **6.2.1 Риски с стороны Frontend**

- Высокие требования к производительности:
  - Пиксель-арт и анимации могут потребовать оптимизации для работы на устройствах с ограниченными ресурсами (например, старых мобильных телефонах).

- **Решение:** Использование Unity's Sprite Packer для оптимизации текстур и уменьшения размера сборки.
- Сложность реализации мини-игры для ловли рыбы:
  - Реализация механик (например, точность подсечки или управление силой заброса) может быть технически сложной.
  - **Решение:** Создание прототипа механики на ранних этапах разработки для тестирования и отладки.
- Проблемы с адаптивностью интерфейса:
  - Разные экраны устройств (мобильные, планшеты, десктопы) могут вызывать проблемы с расположением элементов UI.
  - **Решение:** Использование Canvas Scaler в Unity для автоматической адаптации интерфейса.
- Ошибки при взаимодействии с базой данных:
  - При сохранении прогресса возможны конфликты или потери данных из-за некорректного взаимодействия с SQLite.
  - **Решение:** Добавление проверок целостности данных и аварийного сохранения.

### 6.2.2 Риски с стороны Backend

- Ограничения SQLite:
  - SQLite подходит для локального хранения данных, но может не справиться с большими объемами информации (например, если добавляются новые функции, требующие более сложную структуру БД).
  - **Решение:** если проект станет успешным, можно перейти на более мощную СУБД, такую как PostgreSQL.
- Безопасность сохраненного прогресса:

- Игроки могут попытаться взломать файлы сохранений и изменить свой прогресс (например, увеличить количество денег или уровень).
- **Решение:** Шифрование данных перед сохранением и использование контрольных сумм для проверки целостности файла.
- Сложности при масштабировании:
  - Если игра получит большой успех, текущая архитектура с локальной базой данных может стать недостаточной для обработки множества пользователей.
  - **Решение:** Внедрение облачных решений (например, Firebase или AWS) для глобального хранения данных.
- Ошибки при обновлении контента:
  - Добавление новых видов рыб, локаций или улучшений может привести к несовместимости с существующим кодом.
  - **Решение:** Написание модульного кода и проведение регулярных тестов перед выпуском обновлений.