# Введение

Разработка компьютерных игр является одной из востребованных и быстрорастущих отраслей. Создание игр доступно не только большим студиям, а качественный проект может приносить большой пассивный доход.

Целью проекта является разработка Веб-игры - которая понравиться игрокам и будет в дальнейшем поддерживаться. Игроков должна зацепить идея игры достаточно чтобы им было интересно проводить в ней время.

Поскольку используется WEB разработка - в качестве средств разработки выступает Visual Studio Code и фреймворк Node.js. В качестве платформы для запуска приложения на сервере выступает Docker, а в качестве базы данных PostgreSQL.

Среди практически значимых аспектов результата можно выделить предоставляемый развлекательный контент, завлекающий игроков и позволяющий им весело провести время.

# **Анализ предметной области и формулировка требований.**

Экономические стратегии - это вид стратегий, в котором акцент смещается со сражений на полях битвы между фракциями на внутренние проблемы отдельного города, государства или поселения.

В экономических стратегиях игроки занимаются в первую очередь выстраиванием экономики, строительством инфраструктуры и менеджментом в области решения текущих проблем, создавая необходимые для людей условия труда, заботясь о их всестороннем развитии и благополучии.

Есть много хороших представителей жанра, например, Sid Meier's Civilization – Это серия пошаговых стратегий, в которых игроку предоставляется возможность управлять цивилизацией с античных времен до будущего. Игроки исследуют, развивают свои города, участвуют в дипломатии и ведут войны.

WEB игра была выбрана по причине доступности, браузер есть везде, на телефоне, планшете, компьютере. Это делает игру кроссплатформенной, остаётся только адаптировать её интерфейс для разных устройств для удобства управления.

Сеттинг у игры будет древняя Русь. Игроки берут на себя различные роли, например, ремесленник своими стараниями поддерживает город. Воину предстоит отражать нападения недоброжелателей. Игрокам предстоит улучшать защиту города и действовать сообща чтобы победить противников.

Для работы игры будет использоваться серверная технология node.js, благодаря express будет развернут web сервер на виртуальной машине. Помимо этого будет использоваться реляционная база данный PostgreSQL и резидентная система управления базами данных класса NoSQL – redis.  
Все вместе это будет работать благодаря Docker контейнерам и микросервисной архитектуре.

Требования к веб-игре:

Функциональные:

1. Интерфейс игры должен быть интуитивно понятным и удобным для игроков, чтобы они могли легко управлять своим поселением и взаимодействовать с другими игроками.
2. Игрокам должны быть доступны различные игровые ресурсы, технологии и апгрейды, которые они могут использовать для развития своего поселения.
3. Игра должна иметь возможность многопользовательского режима, чтобы игроки могли взаимодействовать друг с другом, торговать ресурсами, заключать союзы или вести противостояния.
4. Игра должна иметь систему сохранения игрового прогресса, чтобы игроки могли продолжать игру с того момента, где они остановились.
5. Баланс, предотвратить, получение необоснованного преимущества одних игроков, над другими.

Нефункциональные:

1. Игра должна быть оптимизирована для работы на различных устройствах и браузерах, чтобы обеспечить удобство игры для всех игроков.

# Глава 2. Анализ аналогов и прототипов

1. **Forge of empires** — Браузерная многопользовательская игра от InnoGames в жанре экономической стратегии и градостроительного симулятора.  
   **Плюсы**:

* Красивая графика
* Полная свобода действий (строим что хотим и когда хотим)

**Минусы**:

* Приходится уделять игре много времени чтобы достичь результата
* Множественные глюки
* Блокировки учетных записей игроков
* Неудобный обзор сверху

**Функциональные возможности**:

* Настройка пользовательских таймеров
* Древо навыков
* Система веков (все улучшается)
* Донат не обязателен, но присутствует повсеместно
* Тактический режим боя между игроками

**Модель распространения:** Реклама

Присутствуют внутриигровые покупки, влияющие на игровой процесс

1. **Travian** — это браузерная MMORPG, которая появилась в 2004 году.  
   В игре нужно строить свою деревню, защищать её от врагов, собирать ресурсы и готовиться к главному событию — строительству Чуда Света.  
   **Плюсы**:

* Выбор стартовой фракции
* Цикличный игровой процесс позволяющий не затягивать игру
* Приятная для глаза графика

**Минусы**:

* Донат влияет на игровой процесс
* Отсутствует разнообразие тактики

**Функциональные возможности**:

* Присутствует персонаж игрока
* Основной игровой ресурс - время

**Модель распространения: Реклама**

**Присутствуют** внутриигровые покупки, влияющие на игровой процесс

1. **Stronghold Kingdoms** — онлайн игра основанная на игре stronghold про проектирование и строительство замка. Дата выпуска - 2010 год. В основе игры лежит все тот же принцип что и в оригинальном Stronghold но жанр игры изменился.

**Плюсы**:

* Интересная система прокачки
* Древо исследований
* Карта, на которой можно строить новые государства

**Минусы**:

* Медленная прогрессия
* Влияющий на игру донат
* В игре нет ничего нового
* Администрация настроена негативно к игрокам

**Функциональные возможности**:

* Взятый с Stronghold функционал

**Модель распространения:** Реклама

Присутствуют внутриигровые покупки, влияющие на игровой процесс

**Вывод:**

В целом аналоги представляют собой список схожих черт в разной оболочке и в своими небольшими особенностями. Повсеместно присутствуют внутриигровые покупки, влияющие на игровой процесс. Игры в большинстве своем не привносят ничего нового в жанр и слабо отличаются друг от друга. Основным минусом каждой из игр можно вынести внутриигровые покупки влияющие на игровой процесс, данный метод монетизации показывает крайне большую прибыль но не нравится игрокам, так как им приходится сталкиваться с дисбалансом сил в пользу игроков которые вкладывают реальные средства в игру.

# Глава 3. Проектирование и программирование

## Раздел 3.1. Описание интерфейса и архитектуры разрабатываемого продукта

Игра не нагружена графикой, использован “material” интерфейс, используется шрифт Segoe UI т.к. он поддерживается большинством браузеров и выглядит приятно и красиво. В качестве основных элементов интерфейса выступает нижнее меню с кнопками навигации, кнопки могут сменять друг друга в зависимости от того на каком меню находится игрок, переключение между меню как раз и происходит по нажатию на эти кнопки. Какие-то кнопки переводят игрока на другие меню, а другие кнопки позволяют взаимодействовать с игровым миром и например – открыть рынок, выйти в лес, создать предмет.

Внешний вид приложения должен хорошо выглядеть как на компьютере, так и на телефоне, есть адаптация на различные экраны на различных устройствах, кнопки должны выделяться и их цвет должен отличать их по функционалу. Интерфейс не перегружен и отпугивает пользователя огромным количеством функционала, а также должен быстро работает на всех устройствах и не создает дискомфорт во время игры.

Игра должна привлекать игроков в первую очередь своей прогрессией и социальным взаимодействием. Крайне увлекательно достигать все больших успехов и видеть, как эти успехи отражаются на остальных игроках. К примеру можно построить в городе новое здание и наблюдать как им пользуются другие игроки. Проведённое время в игре должно поощряться, игрок который долгое время собирал ресурсы прокачивает навык добычи и собирает ресурсы лучше других игроков. Так же и с другими активностями.

Используется клиент-серверная архитектура, код сервера храниться только на сервере, он недоступен на устройствах клиентов. Все вычисления выполняются на сервере, поэтому требования к устройствам, на которых установлен клиент – снижаются. Все данные хранятся в базе данных, клиенты не имеют к ней прямого доступа.

Сервер находится на международном хостинг провайдере – beget что позволяет не думать о поддержке собственного сервера. База данных работает там же, рядом с сайтом что позволяет минимально сократить время ответа, помимо этого используется система управления базами данных redis которая позволяет работать с кэшем и в разы сокращать время ответа сервера на частые запросы пользователей.

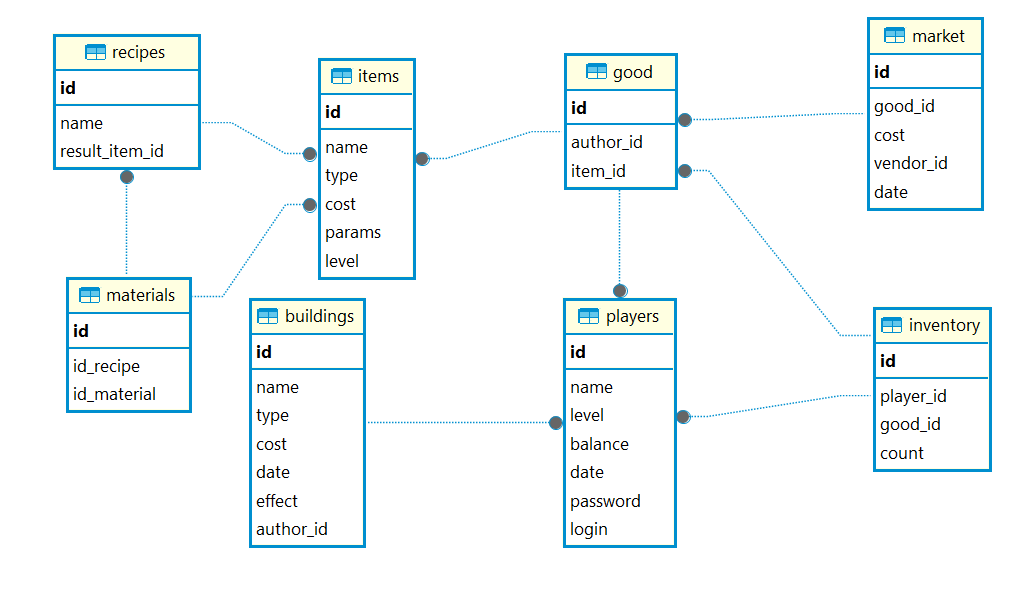
Проект делится на несколько модулей –

* Сервер, все файлы находятся на сервере и доступ к ним происходит только по запросу клиента
* Основной HTML интерфейс, включает в себя также код кнопок которые запускают функции в других модулях
* Главный скрипт – Общие функции которые достаточно малы чтобы не создавать под них отдельный модуль
* Скрипт строительства – Описывает логику строительства построек, логика самих построек работает на сервере
* Скрипт боя – описывает логику боя с противниками ботами

Сервер и клиент связаны между собой технологией socket с помощью js(JavaScript) библиотеки Socket.IO позволяющей обмениваться клиенту и серверу данными в реальном времени. Также присутствует REST API благодаря которому клиент получает главную страницу, доступ к файлу стилей а также со стороны администратора предусмотрены некоторые технические маршруты которые упрощают администрирование.

База данных состоит из нескольких таблиц  
Таблица игроков, предметов, рецептов, инвентаря, рынка, зданий.

У рецепта есть предмет от которого этот рецепт и связанная 1 ко многим таблица материалов, у рецепта может быть множество материалов. Материалы являются таблицей предметов, таким образом возможны многокомпонентные структуры, в которых один предмет состоит из нескольких попроще. Также есть таблица good(Товар) которая олицетворяет собой физическое воплощение предмета. Товар может быть в инвентаре или в маркете. Также у товара может быть автор, это таблица player(Игрок). Игрок может строить здания и создавать предметы тем самым становясь их автором.

**

*Схема базы данных*

## Раздел 3.2. Разработка внутренней структуры и программирование

У проекта хорошая файловая структура, в корне проекта присутствуют файлы index.js и основные файлы сервера. Package.json для правильной работы node.js и npm, dockerfile с инструкциями для Docker для правильного создания образа проекта. также здесь расположены модули с кодом различных частей сервера для лучшей архитектуры кода.

Присутствует папка public в которой хранятся все файлы, которые доступны пользователю, среди них index.html, style.css и script.js. Все эти файлы необходимы на стороне пользователя для отображения информации на экране и красивого изображения, в html разметке присутствуют все элементы интерфейса, которые видит пользователь, в css стилях для этих элементов указаны специальные опции для их красивого отображения на экране, и их изменение в зависимости от экрана пользователя, таким образом изображение на телефоне и компьютере будет отличаться. Также на стороне пользователе присутствует папка files и папка images, там находятся картинки к которым имеет доступ пользователь и которые отображаются на экране.

Средствами программирования являются редактор visual studio code, для управления базой данных используется DBeaver, для управления контейнерами используется Docker, в качестве инструмента по управлению сервером используется Windows PowerShell и подключение по ssh протоколу.

Для создания Веб-сервера используется фреймворк express для node.js, он предоставляет набор инструментов для создания маршрутов, обработки запросов и ответов, управления сессиями и некоторые другие функции. Ниже представлен код получения главной страницы нашего веб приложения.

app.use(express.static('public'));

app.get("/", (req, res) => {

    if (!req.body.token) {

        res.redirect("/login");

    }

    res.sendFile(path.join(\_\_dirname, "index.html"));

});

В данном коде мы подключаем папку public для того чтобы доступ к файлам в ней был открыт для всех пользователей, это позволяет не прописывать отдельно путь до каждого изображения и не нагружать этим код. При обращении к корню сайта мы проверяем вошёл ли пользователь в аккаунт. Если же у него есть токен то он будет расшифрован и пользователь получит данные о своем состоянии, своем инвентаре и остальную информацию. Если же у игрока нет токена то его переадресует на страницу авторизации, на ней же есть ссылка на страницу регистрации. После авторизации/регистрации пользователю выдаётся jwt токен с необходимыми данными для входа и происходит переадресация на корень. Сервер постоянно делает запросы к базе данных для получения искомой информации, в качестве базы данных используется postgresql, для подключения использвется библиотека pg.js. Подключение к базе данных реализовано отдельным файлом в виде следующего кода:

const { Pool } = require('pg');

const pool = new Pool({

    host:'45.146.165.71',

    port:5438,

    user: "postgres",

    password: "\*\*\*\*\*\*\*\*",

    database: "rusonomic"

})

module.exports = pool;

Опиши весь процесс обработки данных –Что происходит после нажатия на кнопку в интерфейсе? Как устроен сервер? Как клиент обменивается с сервером информацией? Какие основные функции используются в коде? Какие самые значимые элементы в игре? Как они работают? \*Блоксхема, алгоритм работы приложения\* с описанием, примеры отдельных форм с описанием программного кода отдельных кнопок, подробно описывается одна из форм, комментарии к коду.