МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Скриптовые языки программирования»

Тема «Настольная игра»

**Исполнитель**

студент 2 курса 7 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ковалев А.Л.

подпись, дата

**Руководитель**

Доцент каф. ИСиТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жиляк Н. А.

должность, учен. степень, ученое звание подпись, дата

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жиляк Н. А.

подпись дата инициалы и фамилия

**Содержание**

[Введение 5](#_Toc121919361)

[1. Постановка задачи 6](#_Toc121919362)

[1.1 Обзор аналогичных решений 6](#_Toc121919363)

[1.2 Техническое задание 7](#_Toc121919364)

[1.3 Выбор средств реализации программного средства 8](#_Toc121919365)

[1.4 Вывод 10](#_Toc121919366)

[2. Проектирование страниц веб-сайта 11](#_Toc121919367)

[2.1. Выбор способа верстки 11](#_Toc121919368)

[2.2. Выбор стилевого оформления 12](#_Toc121919369)

[2.3. Выбор шрифтового оформления 13](#_Toc121919370)

[2.4. Разработка логотипа 14](#_Toc121919371)

[2.5. Разработка пользовательских элементов 15](#_Toc121919372)

[2.7. Выводы 16](#_Toc121919373)

[3. Реализация структуры веб-сайта 17](#_Toc121919374)

[3.1. Настройка Webpack и Babel 17](#_Toc121919375)

[3.2. Структура JSX-документа 19](#_Toc121919376)

[3.3. Структура Redux-Toolkit 20](#_Toc121919377)

[3.4. Использование языка TypeScript 21](#_Toc121919378)

[3.5. Использование стандартов SVG 22](#_Toc121919379)

[3.4. Использование платформы Node.js 23](#_Toc121919380)

[3.5. Выводы 24](#_Toc121919381)

[4 Тестирование веб-сайта 25](#_Toc121919382)

[4.1. Адаптивный дизайн веб-сайта 25](#_Toc121919383)

[4.2. Кроссбраузерность веб-сайта 26](#_Toc121919384)

[4.3. Выводы 28](#_Toc121919385)

[Заключение 29](#_Toc121919386)

[Список использованных источников 30](#_Toc121919387)

# 

# Введение

В настоящее время Интернет становиться все более развитой средой для осуществления коммуникаций. В связи с глобальным развитием сети Интернет, в программировании все более резко начала выделяться отдельная его отрасль — веб-программирование.

Сейчас, чтобы привлечь внимание клиентов, покупателей или партнёров, просто необходимо заявить о себе в интернете, путём создания веб-сайта. Эффективно работающий ресурс положительно влияет на уровень продаж компании, укрепляет ее позиции среди конкурентов. Поэтому роль сайта чрезвычайно велика, и от того, насколько правильно он подобран, во многом зависит успешность бизнеса.

Тема моей курсовой работы «Настольная игра». Веб-сайт предоставляет собой копию известной настольной игры - монополия.

Целевая аудитория игр это молодые люди в возрасте 15-30 лет, но при этом мы не будет ограничится в данной выборке. Но при этом из-за того, что она появилась недавно, нет еще устоявшихся принципов, и каждый может показать себя в ней. Это те причины, из-за которых я выбрал именно эту тему для курсовой работы.

Основная цель курсового проекта – развитие навыков вёрстки сайтов, используя технологии React & Redux. В ходе выполнения курсовой работы будет получен опыт в разработке и кроссбраузерных веб-сайтов, создании анимации и переходов, используя JavaScript навыки.

Задачи данной курсовой работы: реализовать кроссбраузерность веб-сайта; подобрать уникальный дизайн веб-сайта, который будет выделять его среди конкурентов и создавать комфортные ощущения у пользователя. Для этого нужно исследовать несколько популярных веб-сайтов этой же темы и выделить основные вещи, которые будут реализованы и у моего курсового проекта.

# 1. Постановка задачи

## 1.1 Обзор аналогичных решений

Прежде, чем начать разработку собственного игры следует исследовать сайты с подобной тематикой. Исследование сайтов даст нам понятие того, как следует распределять элементы на странице, какие технические решения прижились, а какие элементы дизайна лучше следует избегать.



Рисунок 1.1 – Изображение веб-сайта monopolystar.ru

Первый сайт, который я рассмотрел это оказался сайт monopolystar.ru, увидеть его можно на рисунке 1.1. К плюсам можно отнести внешний вид сайта, сделанный под металлический оттенок. не перегружен элементами, из-за этого интерфейс понятен, в нем. Но в некоторых местах дизайн выглядит непонятно и местами очень старо. Расположение элементов не всегда логичное, и может принести некоторое неудобство при использовании.



Рисунок 1.2 – сайт игры ru.monopoly-club.com

Второй рассматриваемый продукт — это еще один клон игры монополии. В отличие от прошлого сайта тут дизайн уже более современный и выдержан в одной цветовой схеме и стиле. На сайте так же присутствует довольно много анимаций, которые помогаю пользователю определится на какое место на экране им нужно смотреть.

## 1.2 Техническое задание

Основными требованиями к программному продукту являются кроссбраузерность, лёгкость. На создаваемом сайте не должно быть большое количество комплексных программных функций или очень сложных анимаций – всё это может сильно замедлить скорость загрузки информации и ухудшить её восприятие.

Данный продукт должен реализовывать те функции, которые являются основными функциями настольной игры монополия: заход в игру; Просмотр текущей игры; Покупка компаний; подгрузка информации в случае запуска игры; Страницы не должны быть перегружены информацией, должен преследоваться во всем дизайне единый стиль.

Постановка задач

Основными задачами курсового проекта являются:

* Создание сайта по технологии spa(singe page application);
* Разработка страниц сайта для разных браузеров;
* Создание уникального дизайна;
* Изучение новых технологий.

Основными задачами программного средства являются:

* Отображение игры;
* Реализация динамической генерации контента из базы;
* Предоставление информации об команиях;
* Возможность оставлять сообщения в чат;
* Возможность узнать текущий ход;

## 1.3 Выбор средств реализации программного средства

Сайт разрабатывался в текстовом редакторе Visual Studio Code. Страницы сайта могут просматриваться в таких браузерах как Microsoft Edge, Safari, Opera, Mozilla Firefox, Google Chrome и им подобные.

Visual Studio Code, также часто называемый VS Code, - это редактор исходного кода, созданный компанией Microsoft на базе Electron Framework для Windows, Linux и macOS. Функции включают поддержку отладки, подсветку синтаксиса, интеллектуальное завершение кода, фрагменты, рефакторинг кода и встроенный Git. Пользователи могут менять тему оформления, сочетания клавиш, настройки и устанавливать расширения, добавляющие дополнительную функциональность.

В опросе разработчиков Stack Overflow 2021, Visual Studio Code был признан самым популярным инструментом среды разработчика среди 82 000 респондентов, 70% из которых сообщили, что используют его.

Будет разработка «full stack» приложения, то есть будет как серверная часть приложения, так и клиентская часть.

* На серверной части будет использована среда node js и библиотека socket.io на языке JavaScript;
* На клиентской части будет использован react.js, redux, а так же компиляция с помощью webpack, на языке TypeScript;

Node или Node.js — программная платформа, основанная на движке V8, превращающая JavaScript. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Socket.IO — JavaScript-библиотека для веб-приложений и обмена данными в реальном времени. Состоит из двух частей: клиентской, которая запускается в браузере и серверной для node.js. Оба компонента имеют похожее API. Подобно node.js, Socket.IO событийно-ориентированная.

Socket.IO главным образом использует протокол WebSocket, но если нужно, использует другие технологии, например Flash Socket, AJAX Long Polling, AJAX Multipart Stream, предоставляя тот же самый интерфейс. Помимо того, что Socket.IO может быть использована как оболочка для WebSocket, она содержит много других функций, включая вещание на несколько сокетов, хранение данных, связанных с каждым клиентом, и асинхронный ввод/вывод.

При разработке веб-сайта использовались такие языки как: TypeScript JavaScript.

JavaScript – это интерпретируемый язык, то есть исходный код скриптов не нуждается в компиляции. Интерпретаторы этого языка автоматически управляют памятью и очищают неиспользуемые блоки, следовательно, программисту не нужно следить за выделением и очисткой памяти, как, например, в C++. Синтаксис JavaScript был вдохновлён языками C и Java. Все идентификаторы (названия переменных, функций и т.д.) являются регистрозависимыми. Операторы разделяются при помощи точки с запятой «;», однако использование этого символа необязательно: оператор автоматически завершается в конце строки.

React (также известный как React.js или ReactJS) — это бесплатная библиотека с открытым исходным кодом на JavaScript для построения пользовательских интерфейсов на основе компонентов пользовательского интерфейса (UI). Она поддерживается компанией Meta (ранее Facebook) и сообществом индивидуальных разработчиков и компаний. React может быть использован в качестве основы при разработке одностраничных, мобильных или серверных рендеринговых приложений с помощью таких фреймворков, как Next.js. Однако React занимается только управлением состоянием и рендерингом этого состояния в DOM, поэтому создание приложений React обычно требует использования дополнительных библиотек для маршрутизации, а также определенной функциональности на стороне клиента.

TypeScript — язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript

Спецификации языка открыты и опубликованы в рамках соглашения Open Web Foundation Specification Agreement (OWFa 1.0).

TypeScript является обратно совместимым с JavaScript и компилируется в последний. Фактически, после компиляции программу на TypeScript можно выполнять в любом современном браузере или использовать совместно с серверной платформой Node.js. Код экспериментального компилятора, транслирующего TypeScript в JavaScript, распространяется под лицензией Apache. Его разработка ведётся в публичном репозитории через сервис GitHub.

TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ.

Планируется, что в силу полной обратной совместимости адаптация существующих приложений на новый язык программирования может происходить поэтапно, путём постепенного определения типов.

На момент релиза представлены файлы для восприятия расширенного синтаксиса TypeScript для текстового редактора кода Visual studio Code.

Redux - это контейнер предсказуемого состояния для приложений JavaScript.

Он помогает писать приложения, которые ведут себя последовательно, работают в различных средах (клиентской, серверной и нативной) и легко тестируются. Кроме того, он обеспечивает отличные возможности для разработчиков, такие как редактирование кода в реальном времени в сочетании с отладчиком, перемещающимся во времени.

Redux можно использовать с React или с любой другой библиотекой представлений. Он небольшой (2 кБ, включая зависимости), но имеет большую экосистему доступных аддонов.

## 1.4 Вывод

В результате анализа существующих веб-сайтов был выявлен основной функционал программного продукта и поставлены задачи, которые необходимо решить. Выбранный стек технологий позволяет реализовать общую задачу по разработке веб-сайта, и, при этом, решить такие задачи, как адаптивность и динамическая генерация контента.

# 

# 2. Проектирование страниц веб-сайта

## 2.1. Выбор способа верстки

В данном курсовом проекте я выбрал современные технологии, а именно «grid-верстку», которая во всех современных браузерах, она имеет неоспоримые преимущества над другими в том, что она позволяет располагать контент вдоль двух направлений осей. Поддержку grid можно увидеть на рисунке 2.1.

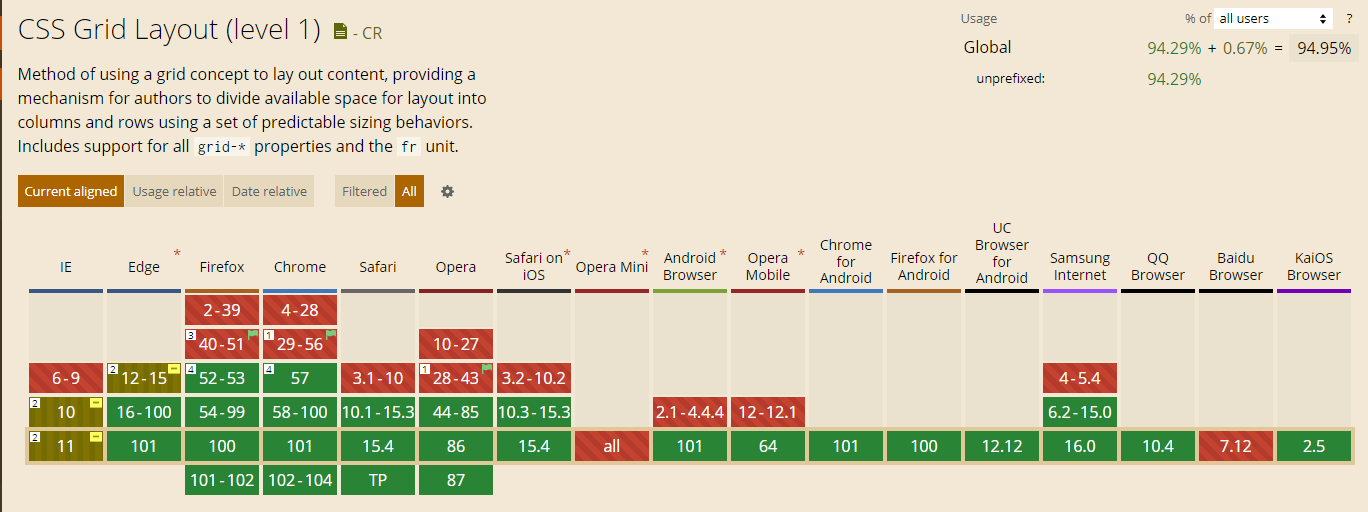


Рисунок 2.1 – Информация с сайта caniuse.com, показывающая поддержку grid

К преимуществам grid можно отнести

* Экономия времени на сложных макетах.
* Лучшая производительность по сравнению с аналогами.
* Лучшая поддерживаемость браузерами по сравнению с аналогами.
* Имеет больше возможностей чем аналоги.

В данной курсовой работе так же я использовал как подспорье другую современную технологию как flexbox, которая позволяет использовать контейнер, который подстраивается под содержимое и это позволяет сократить время на создание сайта. Узнать про поддержу flexbox разными браузерами можно на рисунке 2.2.

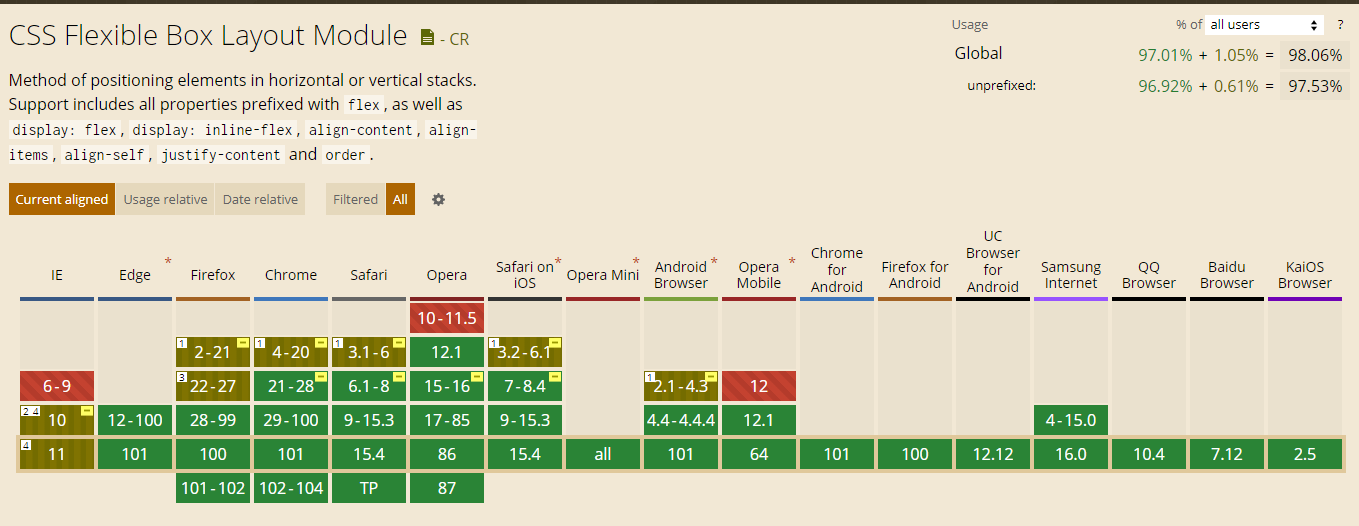


Рисунок 2.2 – Изображение с сайта caniuse.com, показывающая поддержку flexbox

К преимуществам flexbox можно отнести

* Блоки гибкие, элементы могут сжиматься и растягиваться, заполняя пространство.
* Элементы легко можно выровнять по вертикали и горизонтали.
* Изменить порядок элементов можно не только через HTML, но и CSS.
* Простое преобразование и ряда в столбец.
* Очень простая адаптация под направление текста RTL (right-to-left).

Структура сайта к который я буду стремится:

Сайт должен корректно отображаться в основных браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari)

## 2.2. Выбор стилевого оформления

Выбор правильного дизайна для сайта — это очень важный шаг, так как из-за него зависит продолжит ли пользователь пользоваться нашим сайтом или решить перейти на другой. Основные требованию к дизайну такие

* логическая структура страниц;
* эргономичность или по-другому «юзабилити» сайта;
* художественное оформление сайта (сюда входят размер и вид шрифта, картинки, общий фон и стиль и т.д.).

С данными требованиями придётся создать сайт, отвечающий современным стандартам и стеку технологий. И вот какие ключевые решения я выбрал.

Цветовая гамма: Тёмная тема, с разными цветами игроков. Данный выбор был сделал в соответствие с современными тенденциями, что способствует его продвижению. Цветовая тема представлена на изображении 2.3 с цветами, представленными в формате HEX и названия данных цветов.



Рисунок 2.3 – Использованная цветовая палитра

Так же будет присутствовать большое количество закруглений у блоков, что отвечает современным тенденциям веб дизайна. Такие проявления мы можем увидеть на всех современных сайтах таких как: «youtube.com», «vk.com», «google.com» и другие.

Формат используемых изображений: PNG, SVG. Это способствует лучшему отображению сайта в разных браузерах.

## 2.3. Выбор шрифтового оформления

Выбор шрифта — это очень важный шаг, ведь текст — один из основных способов коммуникации дизайнера с пользователями. Так типографика может как улучшить дизайн, так и сломать его.

Из этих соображений я выбрал шрифт «Lato», так как он имеет строгий вид, но при этом не настолько как «Times New Roman». В нем отсутствуют засечки, но при этом он выдержан в строгих линиях что позволяет его использовать в сайтах с серьезной тематикой, что мы может и увидеть по количеству людей, использующих его.

Данный шрифт распространяется бесплатно, поэтому мы его можешь использовать без опасений об авторских прав. Шрифт поддерживает как английское, так и русское начертание, что позволяет нам его использовать как основной во всем курсовом проекте. Шрифт «Lato» можно рассмотреть на рисунке 2.4, он предоставлен с примерами использования на разных языках.

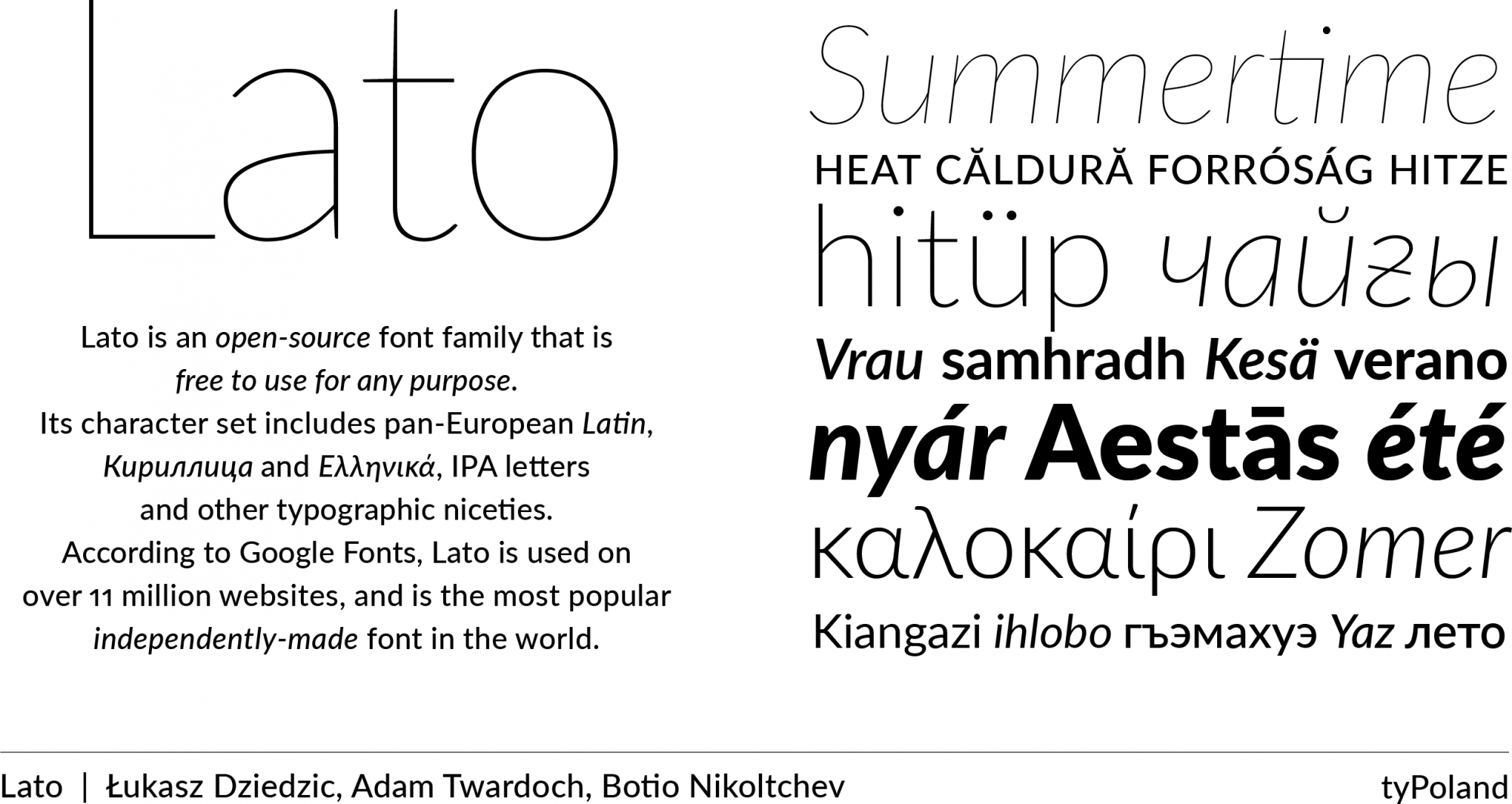


Рисунок 2.4 – Внешний вид шрифта «Lato» в разных начертаниях

Данный шрифт был выбран в сравнении с аналогичными, сравнивали по таким критериям как: строгость шрифта, поддержка русского языка, внешний вид. В данном сравнении победил представленный выше шрифт.

## 2.4. Разработка логотипа

Было сделано решение разрабатывать логотип в стиле минимализм, вдохновляясь мировыми брендами таких как «Meta», «Vk», «Google chrome» и им похожие.

Логотип — абсолютная основа визуализации бренда. Это важная часть рекламного стиля компании, ее можно определить по всем визуальным элементам бренда. Это позволяет отличаться от конкурентов и создавать умы своих клиентов, деловых партнеров и публики с помощью ярлыка продукта или конкретного чувства.

Логотип создавался из простых фигур: параллельных линий, представленных в виде буквы. Логотип отсылает нас на название игры – монополия, но при этом состоит из двух повторяющихся фигур, что позволяет мозгу быстро воспринимать образ, что положительно скажется на запоминании разработанной игры. Формат хранения иконки был выбран SVG, что позволяет нам использовать изображение при разных обстоятельствах и не быть обременённым сложностью, связанной с качеством изображения.

Разработанный логотип предоставлен на изображении 2.5

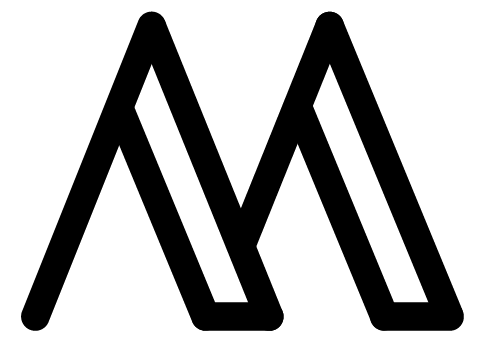


Рисунок 2.5 – Разработанный логотип

Разработка была проведена в программе для работы с SVG под названием «Adobe Illustrator». В нем были сделаны несколько прототипов будущего логотипа и в ходе обдумывания лучшего варианта был выбран представленный выше вариант.

## 2.5. Разработка пользовательских элементов

Поле для настольной игры представлено на рисунке 2.6. Данный макет будет являться ориентиром при разработке игры, но можно будет делать незначительные корректировки в его дизайне.

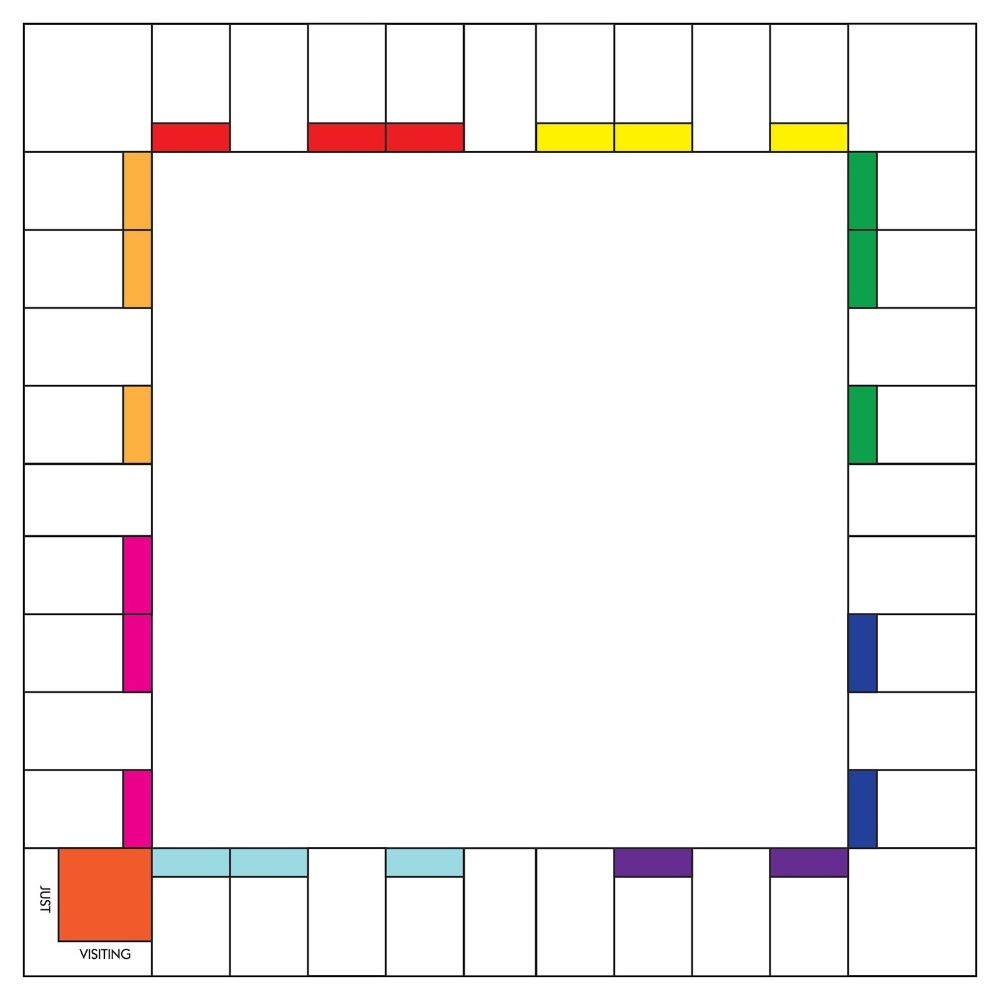


Рисунок 2.6 – Макет поля для игры

На рисунке 2.7 представлены дизайн и цвета, выбранные для игроков, каждый цвет отличается, так что их можно явно разделить между собой. В ходе работы так же будет добавлены отличительные возможности для каждого игрока, что может позволить легче узнавать чей сейчас ход.

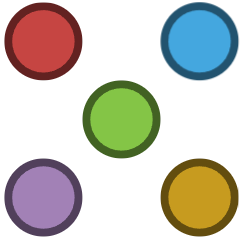


Рисунок 2.7 – Макет игрока

Макет игрока был выбран в виде простого круга, для того что бы была возможность сделать его из обычного HTML элемента, что позволит быстрее загружать сайт, в сравнении, с под грузкой картинки.

## 2.7. Выводы

Исходя из поставленных задач был разработан дизайн в минималистическом виде, приятный и удобный для пользователя. Разметка при помощи системы «Grid» и «flexbox» позволила решить поставленную задачу по обеспечению адаптивности страницы.

На данном этапе были созданы прототипы для облегчения последующей верстки. Было упорядочено расположение всех элементов на страницах было определено стилевое оформление сайта, а именно основная цветовая гамма, шрифтовое оформление, элементы пользовательского интерфейса.

# 3. Реализация структуры веб-сайта

## 3.1. Настройка Webpack и Babel

Из-за того что react является надстройкой на JavaScript с своим синтаксисом, ему требуется компиляция обратно в нативный JS, с этой целью был выбран продукт Webpack и babel, позволяющий сделать это, но не ограничиваясь этим. Он так же позволяет удалить неиспользуемый код («tree shaking»), подключать css как модуль, обрабатывать HTML файлы, и файлы изображений и многое другое.

module.exports = {

mode: "development",

entry: "./src/index.tsx",

output: {

path: path.join(\_\_dirname, "build"),

filename: "bundle.js",

},

resolve: {

extensions: [".js", ".json", ".ts", ".jsx", ".tsx"],

alias: {

'@': path.resolve(\_\_dirname, 'src'),

'@components': path.resolve(\_\_dirname, 'src/components'),

'@css': path.resolve(\_\_dirname, 'src/resources/css'),

'@resources': path.resolve(\_\_dirname, 'src/resources'),

},

},

Листинг 3.1 – настройка выходных параметров

В листинге 3.1 показана настройка Webpack для того, как он будет обрабатывать входные и выходные файлы. Свойство entry указывает откуда начинать сканирование файловой структуры. Следующие свойство указывает куда и как называть выходные файлы. В следующем разделе указаны, какие типы данных обрабатывать, а какие игнорировать.

И последнее свойство alias позволяет при разработке пользоваться не относительными путями, которые при изменении расположения файла перестают работать, а абсолютные пути, которые будут обрабатываться самим webpack-ом.

module:

{

rules: [{

test: /\.m?js$/,

exclude: /node\_modules/,

use: jsLoaders(),

},

{ test: /\.css$/i, use: ["style-loader", "css-loader"] },

{

test: /\.ts$/i,

exclude: /node\_modules/,

use: jsLoaders("@babel/preset-typescript")

},

{

test: /\.jsx$/i,

exclude: /node\_modules/,

use: jsLoaders(["@babel/preset-react", { runtime: "automatic" }])

},

{

test: /\.tsx$/i,

exclude: /node\_modules/,

use: jsLoaders(["@babel/preset-react", { runtime: "automatic" }], "@babel/preset-typescript")

},

{

test: /\.png/,

type: 'asset/resource'

}

],

},

Листинг 3.2– обработка каждого типа данных

В листинге 3.2 представлены типы данных, и как их будет обрабатывать webpack, и в некоторых случаях babel. Для Babel были установлены следующие зависимости –

* "@babel/core": "^7.20.5";
* "@babel/preset-env": "^7.20.2";
* "@babel/preset-react": "^7.18.6";
* "@babel/preset-typescript": "^7.18.6";

Данные зависимости позволяют компилятору babel обрабатывать react, TypeScript, а так же, добавлять в случае необходимости к существующему коду полифилы.

const jsLoaders = (elem, elem2) => {

const presets = [["@babel/preset-env",

{

"useBuiltIns": "usage",

"corejs": "3.25.2"

}]];

if (elem)

presets.push(elem);

if (elem2)

presets.push(elem2);

return {

loader: "babel-loader",

options: {

presets,

}

}

}

Листинг 3.3– функция, облегчающая работу с зависимостями

В листинге 3.3 представлена функция, использующаяся в листинге 3.2, позволяющая проще добавлять обработку к babel, ее основная суть уменьшить количество шаблонного кода.

## 3.2. Структура JSX-документа

React 18 использует в основном функциональные компоненты для отрисовки веб-страницы. С помощью JSX эти компоненты принимают вид подобный структуре HTML-документа. Данная особенность является основным преимущество react по сравнению с другими библеотеками.

import "@css/app.css"

import Board from "@components/Board"

import socket from "@/Socket"

import { addPlayer, changePosition, PlayerInformation, removePlayer } from "./store/PlayersSlice"

import { useAppDispatch } from "./store/hooks"

import Login from "./components/Login"

import { useEffect } from "react"

export default () => {

const dispatch = useAppDispatch()

useEffect(() => {

socket.on('sync', (arr: [number, PlayerInformation][]) => {

arr.forEach(element => { dispatch(addPlayer(element[1])) });

})

socket.on('PlayerLogin', (player: PlayerInformation) => {

dispatch(addPlayer(player));

})

socket.on("playerDisconnect", (index: number) => {

dispatch(removePlayer(index));

})

socket.on('BoughtCompany', (position: number, playerId: number) => { console.log("company buy ,position:" + position)

})

socket.on('dice', (player: number, among: number) => {

dispatch(changePosition({ player, among }))

})

}, []);

return (

<>

<div className="wrapper">

<div className="board-wrapper">

<Board></Board>

</div>

<Login></Login>

</div>

</>

)

}

Листинг 3.4 – компонент App.tsx

В листинге 3.4 Представлен компонент App.tsx, который является корневым элементом, для данной игры, в нем находится отлов основных событий от сервера, и обработка их. Так же в этом же листинге можно увидеть псевдонимы для подключений файлов и стилей. В представленном коде можно заметить использование функций диспетчеров, для соединения с глобальным хранилищем redux, и внесение нужной информации туда. В листинге используется react fragment что позволяет вернуть несколько фрагментов из одного компонента.

import { useState } from "react";

export function useInput({ type }: { type: string }, label: string): [string, JSX.Element] {

const [value, setValue] = useState("");

const input = <label>{label}<input value={value} onChange={e => setValue(e.target.value)} type={type} /></label>;

return [value, input,];

}

Листинг 3.5 – кастомный хук UseInput.tsx

В листинге 3.5 представлен кастомный хук, позволяющий проще работать с окном ввода. Кастомные хуки- это функции которые не являются компонентами react но могут использовать встроенные react хуки.

## 3.3. Структура Redux-Toolkit

Redux-Toolkit это облегченная версия redux, которая позволяет нам убрать огромное количество шаблонного кода(«боллерплейт»), позволяя писать логику намного проще, компактнее и быстрее. На представлен пример слайса, который позволяет нам работать с частью глобального хранилища.

import { createSlice, PayloadAction } from "@reduxjs/toolkit";

export interface PlayerInformation {

name: string

position: number,

money: number

}

export interface Company {

position: number,

price: number

}

export interface ChangePosition {

among: number,

player: number

}

const initialState: PlayerInformation[] = []

export const PlayersSlice = createSlice({

name: 'players',

initialState,

reducers: {

addPlayer: (state, action: PayloadAction<PlayerInformation>) => {

state.push(action.payload);

},

removePlayer: (state, action: PayloadAction<number>) => {

state.splice(action.payload, 1);

},

changePosition: (state, action: PayloadAction<ChangePosition>) => {

const player = action.payload.player

state[player].position += action.payload.among;

if (state[player].position >= 40)

state[player].position -= 40;

},

resetPlayers: () => initialState

}

})

export const { addPlayer, resetPlayers, changePosition, removePlayer } = PlayersSlice.actions

export default PlayersSlice.reducer

Листинг 3.6 – слайс PlayerSlice.ts

В выше представленном листинге 3.6 происходит разработка слайса, которой будет хранить информацию об игроках, и обновлять ее в случае разных событий. Так же в нем представлены интерфейсы для TypeScript, которые позволяют быстрее разрабатывать приложение, за счет подсказок от текстового редактора и предупреждений от компилятора.

import { useDispatch, useSelector } from 'react-redux'

import type { TypedUseSelectorHook } from 'react-redux'

import type { RootState, AppDispatch } from './store'

export const useAppDispatch: () => AppDispatch = useDispatch

export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector

Листинг 3.7 – хуки из store/hooks.ts

Хуки, представленные на листинге 3.7 являют собой типизированную версию хуков из библиотеки react-redux, которая по своей сути является библиотекой, которая соединяет работу react и redux. Типизированные версии разработаны для того, чтобы работа с TypeScript была более точна и понятна.

## 3.4. Использование языка TypeScript

В данном проекте используется язык TypeScript, что позволяет находить ошибки с типами во время разработки, а не во время тестирования, что позволяет быстрее писать логику.

export interface cardInfo {

type: cellType,

rotation: rotation

companyInfo?: number,

}

export interface cardCompanyInfo {

type: cellType,

rotation: rotation

companyInfo: number,

}

export interface companyInfo {

name: string,

group: number,

price: number,

bail: number,

rent: [number, number, number, number, number, number]

}

export interface group {

color: string

priceHouse: number,

priceHotel: number

}

export enum rotation {

top = "top",

right = "right",

down = "down",

left = "left",

corner = "corner"

}

export enum cellType {

company,

question,

train,

airlines,

start,

jail,

casino,

money

}

Листинг 3.8 – основные интерфейсы логики

На листинге 3.8 представлены представлено несколько интерфейсов для основных элементов, которые импортируется везде, где происходит работа с данными об игре. Так же создано два перечисления – отличительная особенность TypeScript которой нету в ванильном JS. Созданные перечисления используется для сохранения позиции клеток и их тип.

## 3.5. Использование стандартов SVG

В данном курсовом проекте была задействована технология SVG. А именно использовались парные теги <svg> и </svg>. Эта технология позволяет нам использовать такие изображение, которые при изменении экрана не будут ухудшатся.

В данном курсовом проекте для отображения иконок, использоваться SVG-изображения.

<svg id="Layer\_1" data-name="Layer 1" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 465.44 335.59">

<defs>

<style>

.cls-1 {

fill: none;

stroke: #231f20;

stroke-linecap: round;

stroke-miterlimit: 10;

stroke-width: 30px;

}

</style>

</defs>

<line class="cls-1" x1="15" y1="320.53" x2="137.22" y2="15" />

<line class="cls-1" x1="260.89" y1="320.59" x2="138.12" y2="17.21" />

<line class="cls-1" x1="193.76" y1="320.26" x2="104.21" y2="103.8" />

<line class="cls-1" x1="260.89" y1="320.59" x2="194.11" y2="320.59" />

<line class="cls-1" x1="232.97" y1="242.11" x2="324.35" y2="15.15" />

<line class="cls-1" x1="449.24" y1="317.4" x2="324.38" y2="15.14" />

<line class="cls-1" x1="381.75" y1="320.13" x2="291.47" y2="101.55" />

<line class="cls-1" x1="450.44" y1="320.56" x2="381.75" y2="320.56" />

</svg>

Листинг 3.9 – Использование SVG

В куске кода, представленном выше, были использованы разные возможности SVG: тег defs позволяющий нам скрывать содержимое.

## 3.4. Использование платформы Node.js

Node.js это платформа позволяющая писать нам код для сервера, при этом реализующая работу с системой.

const express = require('express');

const app = express();

const http = require('http');

const server = http.createServer(app);

const { Server } = require("socket.io");

const io = new Server(server);

const path = require("path");

const buildPath = path.join(\_\_dirname, '..', 'client/build');

app.use(express.static(buildPath))

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

res.sendFile(path.join(buildPath, '/index.html'));

});

Листинг 3.10 – Использование Express

В листинге 3.10 представлен участок кода, который позволяет быстро и просто развернуть сервер, технология, которая используется – это express. Так же инициализируется экземпляр объекта socket.io, позволяющий нам создать непрерывную связь между клиентом и сервером.

io.on('connection', (socket) => {

let playerId = -1;

let ready = undefined;

if (players.size > 0) {

socket.emit('sync', Array.from(players));

}

const checkPlayerPosition = () => {

const player = players.get(turn);

const type = cells[player.position].type;

if (type === 'company') {

standOnCompany(player, cells[player.position].companyInfo)

}

}

const standOnCompany = (player, company) => {

readyToEndMove = false;

const position = player.position;

if (field[position][0] === -1) {

const price = companiesInfo[company][1];

if (player.money < price) {

socket.emit('aboard', 'not enough money, start auction');

auctionStart(position);

}

nextSocket.emit('BuyingRequest', position, price);

nextSocket.once('RequestResult', (result) => {

if (result) {

io.emit('BoughtCompany', position, playerId);

player.money -= price;

io.emit('MoneyChanged', playerId, player.money);

field[position][0] = playerId;

}

else {

auctionStart(position);

}

})

readyToEndMove = true;

}

}

Листинг 3.11 – Использование socket.io

В листинге 3.11 было использована сучность socket.io, которая позволяет отправлять и получать события от клиента, что позволяет сделать двухстороннюю связь, что необходимо для создания рабочей игры.

## 3.5. Выводы

На данном этапе был полностью написан сайт с помощью React. Была подключена таблица стилей CSS для создания дизайна, корректировки цвета и т.д. Благодаря предыдущим пунктам были созданы многие элементы, были изучены новые технологии. Были реализованы все желаемые требования к проекту, что позволило сайту предоставлять уникальный пользовательский интерфейс. Из необходимых элементов было созданы:

* Создан сайт по технологии spa(singe page application);
* Разработаны страница сайта для разных браузеров;
* Создание уникального дизайна;
* Изучение новых технологий.

# 4 Тестирование веб-сайта

## 4.1. Адаптивный дизайн веб-сайта

Адаптивность создавалось под разные размеры экранов, и основные работы были сделаны в перестановке меню так что бы оно было удобно под любые размеры экрана, примеры представлены на рисунке 4.1-4.2

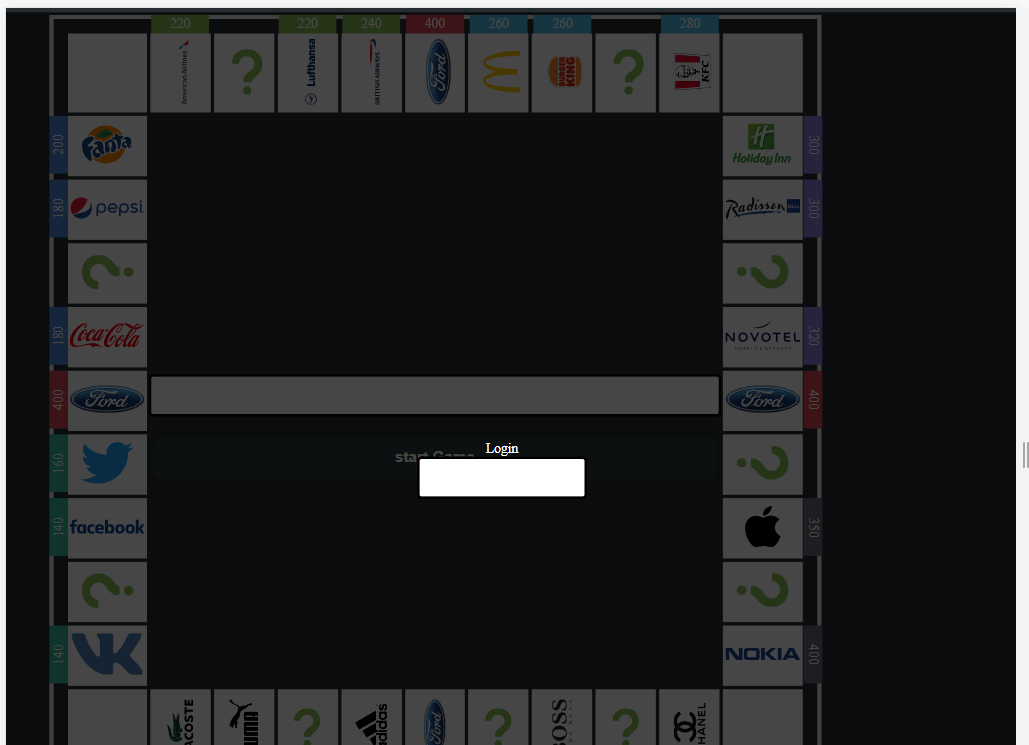


Рисунок 4.1 – Работа сайта в другом разрешении

При кроссплатформенной проверке были выявлены некоторые недочеты с шрифтами и размерами, которые были устранены. Данное тестирование помогло выявить скрытые проблемы сайта.

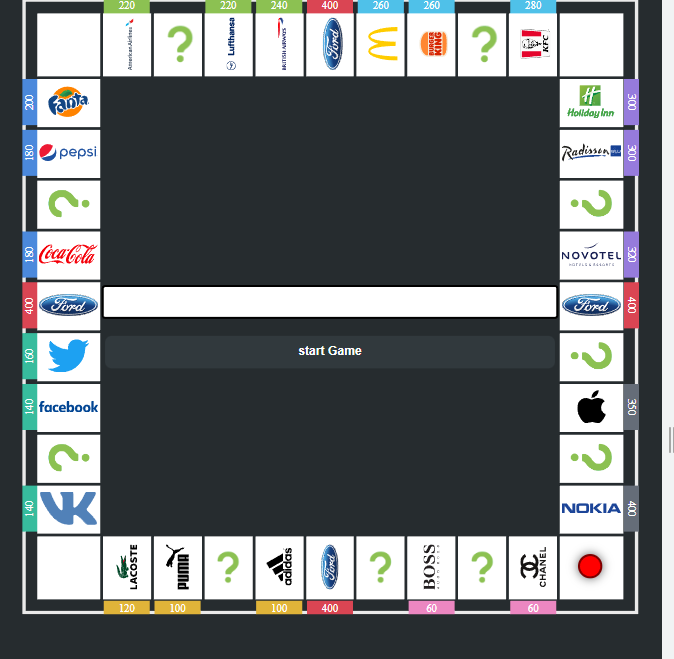


Рисунок 4.2– Работа сайта в мобильном разрешении

## 4.2. Кроссбраузерность веб-сайта

Для проверки, созданной кроссбраузерности веб-сайта, будем использовать такие современные браузеры, как Google Chrome, Microsoft Edge, Brave, Safari, Vivaldi. Пример корректной работы сайта представлен на рисунке 4.3, 4.4, 4.5.

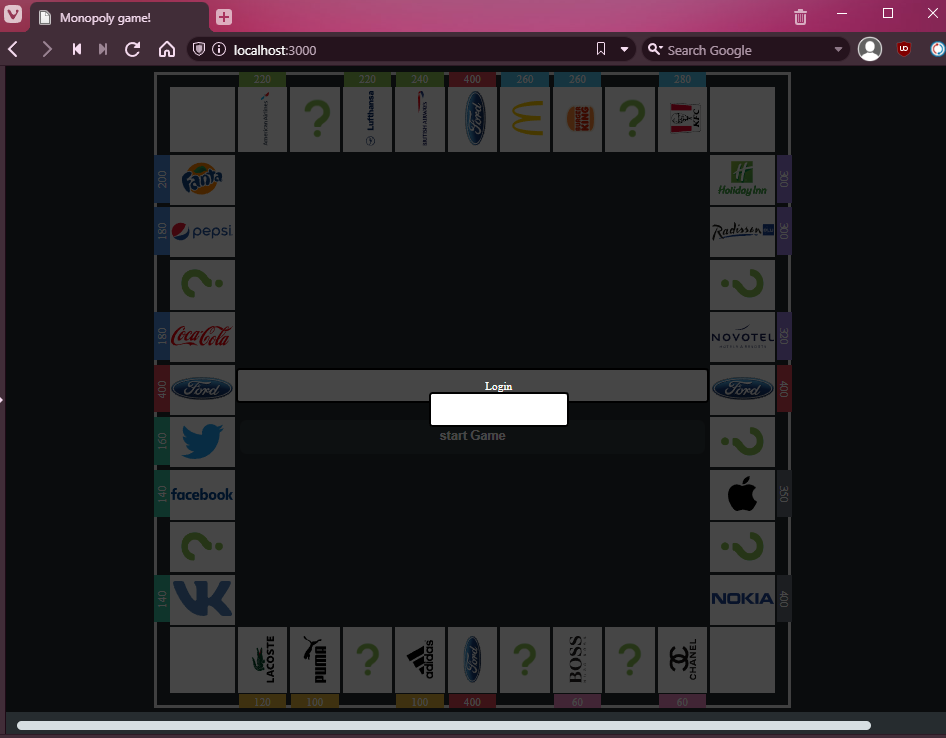


Рисунок 4.4 – Работа сайта в Vivaldi

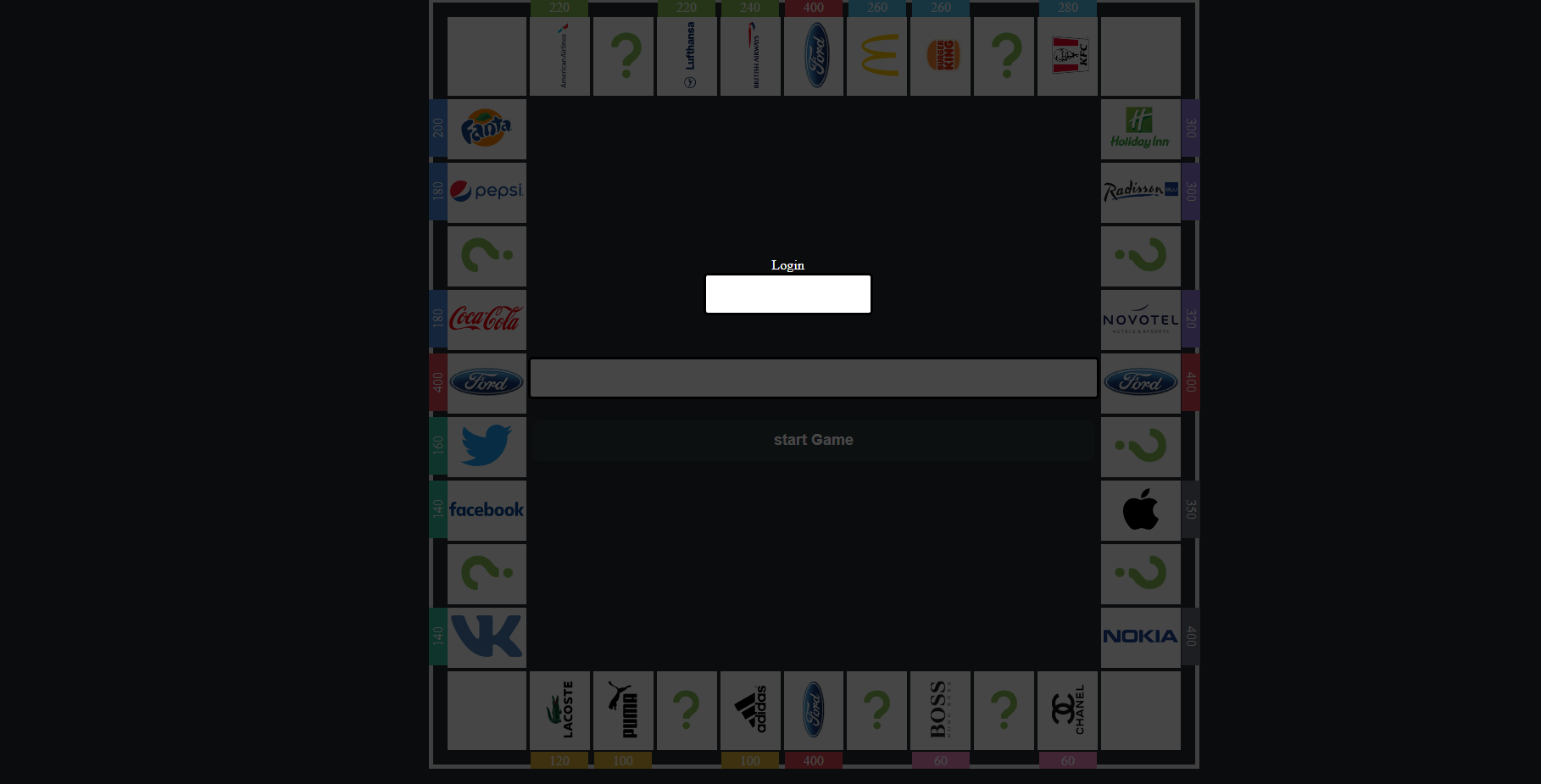


Рисунок 4.5 – Работа сайта в Google Chrome

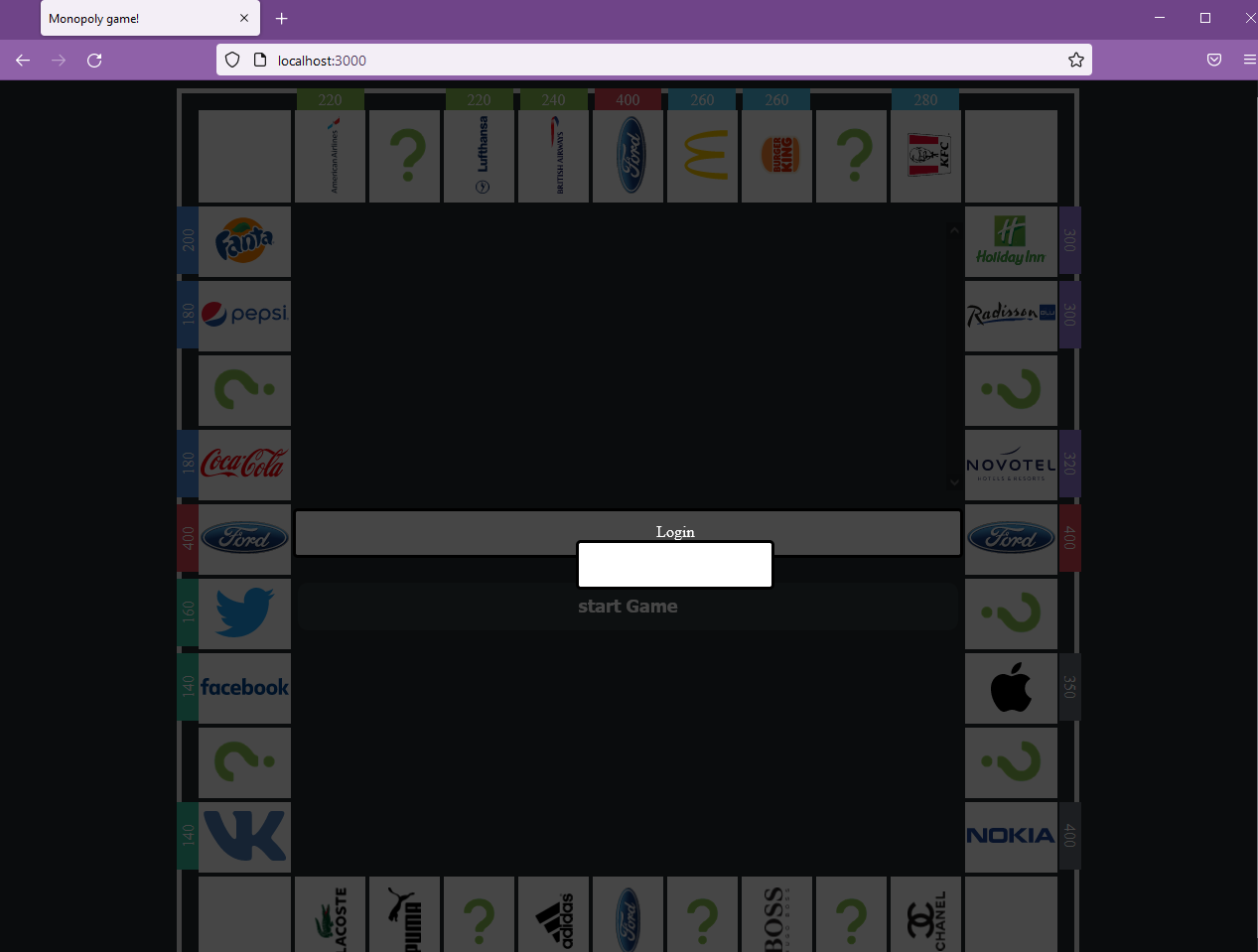


Рисунок 4.6 – Работа сайта в Firefox

Работа сайта в разных браузерах показала себя хорошо, были выявлены незначительные проблемы, решить которые не составило труда.

## 4.3. Выводы

В данном разделе было описано подробное руководство сайтом, которое затрагивает все страницы сайта. Также были проведены кроссбраузерные тесты для проверки работоспособности веб-сайта в разных браузерах, которые помогли выявить проблемы, решение которых помогло улучшить общее качество реализации. Так же были проведены кроссплатформенные тесты, которые показали, что разработанный веб-сайт неплохо адаптирован под современные размеры экранов телефон. Эти тесты позволили улучшить качество всего веб сайта, что положительно повлияет на общее использование сайта разными пользователями.

Во время работы над данным разделом были переделаны некоторые элементы, показавшие себя плохо при кроссбраузерных и кроссплатформенных проверках, из-за этого их нужно было переделать некоторые фрагменты кода, содействующие на общее состояние проекта.

# Заключение

В ходе выполнения данной задачи были рассмотрены аналогичные решения. В процессе осмотра выяснилось, каких тенденции в разработке дизайна веб-сайта стоит придерживаться.

В процессе создания курсового проекта были изучены такие технологии как CSS Flexbox. Были дополнены знания про некоторые возможности react. Узнал много новых CSS-свойств, свойства связанные с SVG, картинками. Были получены практические навыки в разработке полноценного программного продукта, включая прототип, дизайн, адаптивную верстку. Также приобретены новые знания в работе с языком программирования TypeScript. Получен опыт в формировании требований к программному продукту и в прототипировании интерфейсов.

Были разработаны пользовательские элементы, спецэффекты и логотип, для идентификации веб-сайта, среди миллионов других интернет-ресурсов.

Для качественного отображения картинок, на любом устройстве, было решено применять SVG-изображения.

По завершению работы над программным продуктом был создан веб-сайт, отвечающий всем современным требованиям веб-разработки, таких как адаптивность и кроссбраузерность.

При формировании технического задания были продумано содержание основных страниц веб-сайта.

Для разработки были использованы технологии, не изучаемые на данном курсе, такие, как TypeScript, Balel, webpack, socket.io.

# Список использованных источников

1. Просмотр поддержки технологий браузерами [Электронный ресурс] / Справочник Режим доступа: https://caniuse.com– Дата доступа 20.09.2022.
2. Документация по CSS [Электронный ресурс] / Справочник Режим доступа: https://developer.mozilla.org/ru/docs/Web/CSS/ – Дата доступа 21.09.2022.
3. Учебник по JavaScript [Электронный ресурс] / Справочник Режим доступа: https://learn.javascript.ru/ – Дата доступа 23.09.2022.
4. Документация по JavaScript [Электронный ресурс] / Справочник Режим доступа: https://developer.mozilla.org/ru/docs/Web/JavaScript – Дата доступа 23.09.2022.
5. Документация по babel [Электронный ресукрс] / Сайт https://www.babeljs.io Режим доступа: https://babeljs.io/docs/en/babel-preset-react– Дата доступа 23.11.2022
6. Документация по TypeScipt [Электронный ресукрс] / Сайт https://www.typescriptlang.org/ Режим доступа: https://www.typescriptlang.org/tsconfig#jsxImportSource – Дата доступа 23.11.2022
7. Документация по webpack [Электронный ресукрс] / Сайт https://www. webpack.js.org/ Режим доступа: https://webpack.js.org/concepts/module-resolution/– Дата доступа 23.11.2022
8. Документация по node.js [Электронный ресукрс] / Сайт https://nodejs.org Режим доступа: https://nodejs.org/api/timers.html#timeoutrefresh – Дата доступа 23.11.2022
9. Документация по socket.io [Электронный ресукрс] / Сайт https://socket.io/ Режим доступа: https://socket.io/docs/v4/server-api/#event-connectjsxImportSource – Дата доступа 23.11.2022
10. Документация по react [Электронный ресукрс] / Сайт https://reactjs.org/ Режим доступа: https://reactjs.org/docs/refs-and-the-dom.html – Дата доступа 23.11.2022
11. Документация по redux [Электронный ресукрс] / Сайт https://redux-toolkit.js.org/ Режим доступа: https://redux-toolkit.js.org/api/configureStore – Дата доступа 23.11.2022