# **Аннотация**

Данный документ является пояснительной запиской к программному проекту «Онлайн игра “IQ спутник”» - Веб-приложение, предназначенное для онлайн-соревнований по решению головоломки IQ Спутник.

Пояснительная записка содержит следующие разделы: «Введение», «Назначения и область применения», «Технические характеристики», «Ожидаемые технико-экономические показатели» и приложения.

В разделе «Введение» указана краткая характеристика программы, варианты ее названия и документы, на основании которых ведется разработка.

В разделе «Назначения и область применения» указано эксплуатационное и функциональное назначение программного продукта, а также его область применения.

В разделе «Технические характеристики» содержатся следующие подразделы: постановка задачи на разработку программы; описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи и возможные взаимодействия программы с другими программами; описание и обоснование выбора метода организации входных и выходных данных; описание и обоснование выбора состава технических и программных средств.

Настоящий документ разработан в соответствии с требованиями:

1. ГОСТ 19.101-77 Виды программ и программных документов;
2. ГОСТ 19.102-77 Стадии разработки;
3. ГОСТ 19.103-77 Обозначения программ и программных документов;
4. ГОСТ 19.104-78 Основные надписи;
5. ГОСТ 19.105-78 Общие требования к программным документам;
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом;
7. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению.

**Содержание**

[**Аннотация** 2](#_Toc162973433)

[**1. Введение** 4](#_Toc162973434)

[**1.1. Наименование программы** 4](#_Toc162973435)

[**1.2. Основания для разработки** 4](#_Toc162973436)

[**2. Назначение и область применения** 5](#_Toc162973437)

[**2.1. Назначение разработки** 5](#_Toc162973438)

[**2.1.1. Функциональное назначение** 5](#_Toc162973439)

[**2.1.2. Эксплуатационное назначение** 5](#_Toc162973440)

[**2.2. Краткая характеристика области применения** 5](#_Toc162973441)

[**3. Технические характеристики** 6](#_Toc162973442)

[**3.1. Постановка задачи на разработку программы** 6](#_Toc162973443)

[**3.2. Описание и обоснование алгоритма и функционирования программы** 6](#_Toc162973444)

[**3.2.1. Описание алгоритма и функционирования программы** 6](#_Toc162973445)

[**3.2.2. Обоснование выбора алгоритма** 7](#_Toc162973446)

[**3.3. Описание и обоснование метода организации входных и выходных данных** 8](#_Toc162973447)

[**3.4. Описание и обоснование выбора состава технических и программных средств** 8](#_Toc162973448)

[**3.4.1. Состав технических и программных средств** 8](#_Toc162973449)

[**3.4.2. Обоснование выбора состава технических и программных средств** 8](#_Toc162973450)

[**4. Технико-экономические показатели** 9](#_Toc162973451)

[**4.1. Предполагаемая потребность** 9](#_Toc162973452)

[**4.2. Ориентировочная экономическая эффективность** 9](#_Toc162973453)

[**Приложение 1. Список используемой литературы** 10](#_Toc162973454)

[**Приложение 2. Описание и функциональное назначение полей, методов, свойств и классов** 11](#_Toc162973455)

[**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ** 15](#_Toc162973456)

# **1. Введение**

## **1.1. Наименование программы**

Наименование программы – «Онлайн игра "IQ спутник"».

Наименование программы на английском языке – «Online Game "IQ satellite"».

Краткое наименование программы – «IQ спутник».

## **1.2. Основания для разработки**

Разработка ведется на основании учебного плана подготовки бакалавров по

направлению 09.03.04 «Программная инженерия» и утвержденная академическим

руководителем тема курсового проекта.

Программа разрабатывается в рамках выполнения курсового проекта в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия» Национального исследовательского университета «Высшая школа экономики», факультет компьютерных наук, департамент программной инженерии по теме «Онлайн игра "IQ спутник"» под руководством внештатного преподавателя ПИ ФКН Т.В. Каруба.

# **2. Назначение и область применения**

## **2.1. Назначение разработки**

### **2.1.1. Функциональное назначение**

### Приложение предназначено для онлайн-соревнований по решению головоломки “IQ Спутник”. Все игровые механики и взаимодействие реализованы в веб-интерфейсе, без необходимости использования физических компонентов. Пользователи могут зарегистрироваться, создавать и управлять игровыми лобби, а также играть в головоломку в реальном времени с друзьями или другими участниками.

### **2.1.2. Эксплуатационное назначение**

Приложение предназначено для того, чтобы несколько игроков могли одновременно решать головоломку IQ Спутник через веб-браузер, при этом все игровые функции и взаимодействие происходят в браузерной среде, обеспечивая удобный и доступный способ участия в игре.

**2.2.** **Краткая характеристика области применения**

«IQ Спутник» — это многопользовательская онлайн-игра, которая позволяет игрокам соревноваться в реальном времени в решении головоломки. Игра ориентирована на развитие логического мышления и концентрации. Игроки могут создавать игровые лобби, приглашать других участников и вместе решать уникальные головоломки, соревнуясь за лучшее время и точность выполнения заданий.

# **3. Технические характеристики**

## **3.1. Постановка задачи на разработку программы**

Программа обеспечивает выполнение следующих функций:

Игровой функционал:

1. Возможность начать игру
   * Запуск таймера
   * Проверка, что пользователь не собирал головоломку ранее в данном лобби
2. Функционал игрового поля:
   * Отображение игрового поля
   * Рандомизированная генерация поля с поставленной на него деталью
   * Возможность располагать детали на свободные места
   * Проверка собранности головоломки
   * Поиск ближайшего свободного места для вставки детали
3. Функционал игровой детали:
   * Возможность переносить деталь на игровое поле
   * Окрашивание детали в свой уникальный цвет
   * Возможность удалять деталь с игрового поля
   * Возможность поворачивать детали
4. Завершение головоломки
   * Проверка на корректность сборки головоломки
   * Вывод времени, затраченного на сборку

Не игровой функционал:

1. Регистрация пользователя на сайте
   * Заполнение полей для регистрации данными: имя пользователя, логин, пароль
   * Возможность прикрепить фото профиля
   * Выдача JWT токена
   * Переход на главную страницу при успешной регистрации
2. Авторизация пользователя на сайте
   * Заполнение полей для входа: почты и пароля и проверка их корректности
   * Выдача JWT токена
   * Переход на главную страницу при успешной авторизации
3. Поиск игроков по имени и почте
4. Возможность просмотра профиля
   * Должны отображаться имя пользователя, почта, фото профиля
   * Отображение личной статистики: количество игр и среднее время сборки головоломки
5. Создание игровых лобби
   * Поиск игроков по имени и почте
   * Добавление найденных пользователей в лобби
   * Возможность установить имя лобби
   * Возможность выйти из лобби
6. Отображение результатов игроков в лобби
   * Обновление статистики после завершения сборки головоломки пользователем в текущем лобби
7. Система оповещения
   * Оповещение пользователя о приглашении в игровое лобби
   * Оповещение о завершении всеми пользователями головоломки в одном лобби

## **3.2. Описание и обоснование алгоритма и функционирования программы**

### **3.2.1. Описание алгоритма и функционирования программы**

### **Регистрация и авторизация пользователей**

### **Алгоритм**:

### Пользователь вводит регистрационные данные (логин, email, пароль, аватар) через форму на клиентской стороне.

### Клиентская часть на React выполняет валидацию данных

### Данные отправляются на сервер с использованием HTTP-запроса Axios через REST API, созданный с помощью Express.js.

### Сервер с помощью соответствующего контроллера сохраняет данные о пользователе в базе данных MongoDB. Аватар пользователя сохраняется в облачном хранилище cloudinaty, а в базе данных хранится ссылка на этот объект.

### В случае успешной регистрации сервер генерирует JWT-токен и отправляет его на клиент.

### При авторизации введенные данные проверяются по базе данных. Если проверка успешна, сервер также генерирует JWT-токен.

### Токен, персональные данные пользователя и личная статистика сохраняются в локальном хранилище на клиенте, а также в context Storage Express.

### Токен используется для всех последующих запросов, кроме авторизации и регистрации.

### **Обоснование**:

### Использование JWT-токенов позволяет не сохранять информацию о сессии на сервере, повышая безопасность и уменьшая нагрузку на сервер.

### Сохранение на локальном хранилище позволяет взаимодействовать с данными пользователя, не обращаясь к бд. Сохранение в context storage позволяет использовать данные пользователя из различных точек кода программы. Библиотека Axios позволяет легко взаимодействовать с сервером без перезагрузки страницы.

### Валидация данных на клиентской стороне улучшает пользовательский опыт, сокращая количество ошибок ввода.

### **Создание и управление лобби**

### **Алгоритм**:

### После авторизации пользователь может создавать лобби. Для этого он вводит необходимые данные для поиска игроков.

### Данные отправляются на сервер используя Axios, где в контроллере реализован поиск по регулярному выражению, используя Express

### Полученный список пользователей возвращается на клиент. И среди полученных пользователей есть возможность выбора нужных людей в игровое лобби.

### Пользователь может установить название группы или установится имя по умолчанию.

### Данные отправляются на сервер через POST-запрос. Сервер создает запись о новом лобби в базе данных MongoDB.

### Созданное лобби отображается для каждого выбранного пользователя при создании в их списке игровых сессий.

### **Обоснование**:

### Использование regex выражений упрощает и ускоряет код при поиске, а также позволяет работать с большими массивами данных.

### Использование MongoDB позволяет гибко хранить информацию о лобби и игроках в документной структуре. Это удобно, так как количество игроков или параметры могут изменяться динамически.

### Express.js обеспечивает высокую производительность API для обработки запросов от игроков.

### **Игровой процесс: взаимодействие с игровым полем**

### **Алгоритм**:

### После начала игры пользователи получают на клиентской стороне доступ к игровому полю, состоящему из ячеек. Они могут перетаскивать фигуры на поле, используя drag-and-drop.

### Взаимодействие с игровым полем происходит через события, обрабатываемые в React-компонентах.

### Программа проверяет корректность действий игрока: фигура должна находиться в пределах поля и не пересекаться с другими фигурами. Если размещение корректное, обновляется состояние поля.

### Когда собрана полностью головоломка, игра заканчивается и пользователю показываются все результаты игроков в лобби

### Клиентская часть отправляет запрос на сервер с указанием об окончании игры и времени, затраченном на ее сборку. Также обновляется личная статистика в локальном хранилище.

### Сервер синхронизирует данные между участниками лобби.

### **Обоснование**:

### Использование React для работы с состоянием игрового поля позволяет эффективно обновлять интерфейс и отслеживать изменения в реальном времени.

### Обновление данных в локальном хранилище позволяет получать информацию о профиле без обращения к базе данных

### События drag-and-drop делают процесс размещения фигур интуитивным для пользователя.

### Сохранение состояния игры в MongoDB позволяет восстановить прогресс или анализировать результаты.

### **3.2.2. Обоснование выбора архитектуры**

Архитектура MERN выбрана благодаря своей гибкости, производительности и единому языку программирования для всех компонентов (JavaScript). Это упрощает разработку и поддержку приложения, обеспечивая при этом высокую масштабируемость и производительность для работы с многопользовательскими играми и интерактивными интерфейсами.

### **Асинхронность и эффективность обработки запросов**

### Серверная часть, основанная на Express + Node.js, использует асинхронную модель работы. Это позволяет одновременно обрабатывать множество запросов без блокировки основного потока. Для многопользовательской игры это критично, поскольку каждый игрок может отправлять запросы к серверу практически одновременно.

### **Гибкость структуры данных**

### Использование MongoDB позволяет хранить игровые данные, информацию о пользователях и лобби в гибком документном формате, что упрощает организацию сложных и динамичных данных без необходимости жестко определенных схем. Это также способствует быстрому развитию и расширению функционала программы.

### **Безопасность и масштабируемость**

### Применение JWT-токенов для аутентификации обеспечивает высокий уровень безопасности и удобство при масштабировании. Серверу не нужно хранить сессии пользователей, а все данные передаются в зашифрованном виде.

### **Интерактивность и отзывчивость интерфейса**

### Использование React обеспечивает высокую интерактивность клиентской части, что важно для игры с множеством взаимодействий и динамическим изменением состояния (например, при перетаскивании фигур на поле).

## **3.3. Описание и обоснование метода организации входных и выходных данных**

**Входные данные**

**Ввод данных с клиентской стороны (React)**

* + Пользователь взаимодействует с интерфейсом, созданным на React, и вводит данные на различных страницах (регистрация, авторизация, взаимодействие с игровым полем и другие действия). Все данные, вводимые пользователем, валидируются на клиенте перед отправкой на сервер.
  + **React** обеспечивает высокую интерактивность пользовательского интерфейса, что делает ввод данных интуитивно понятным для пользователя.

**Организация и передача данных с помощью API (Express)**

* + После валидации на клиентской стороне данные отправляются через HTTP-запросы (REST API) на сервер, который работает на базе **Express**. Это относится к данным регистрации, авторизации, игровым действиям и взаимодействию с лобби.
  + **Express** как серверная часть является легковесной, быстрой и предоставляет мощный инструмент для создания REST API, что упрощает обработку входных данных.
  + Входные данные в виде JSON передаются через HTTP-запросы, что поддерживает стандартизацию и обеспечивает совместимость с различными фронтенд и бекенд технологиями.

**Выходные данные**

**Обработка и хранение данных в базе данных**

* + После получения данных сервер обрабатывает их и сохраняет в базу данных MongoDB.
  + **MongoDB** является документно-ориентированной NoSQL базой данных, которая идеально подходит для хранения динамичных данных, таких как конфигурация игр и состояние пользователя. Она позволяет гибко сохранять сложные структуры данных без необходимости жесткого соблюдения схем.
  + Высокая производительность при масштабировании и поддержка горизонтального шардирования делает MongoDB идеальной для многопользовательских игр с высокими нагрузками.
  + Встроенная поддержка JSON-формата в MongoDB упрощает обмен данными между клиентом и сервером, поскольку на всех уровнях используется один и тот же формат данных.
  + на всех уровнях используется один и тот же формат данных.

## **3.4. Описание и обоснование выбора состава технических и программных средств**

### **3.4.1. Состав технических и программных средств**

Для нормального функционирования программы требуется компьютер, оснащенный следующими техническими компонентами:

* **Процессор**: не менее 2-х ядер, с тактовой частотой 1.5 ГГц и выше (для настольных и мобильных устройств).
* **Оперативная память**: не менее 2 ГБ для настольных и мобильных устройств, с рекомендованным объемом 4 ГБ для оптимальной производительности.
* **Место на диске**: Приложение не требует установки, но браузер должен иметь не менее 100 МБ свободного места для кэша и хранения данных.
* **Подключение к сети**: Приложение работает в онлайн режиме и требует постоянного интернет-соединения.
* **Периферийные устройства**: Клавиатура и мышь.

Для нормального функционирования программы требуется компьютер, оснащенный следующими программными компонентами:

* **Поддерживаемые браузеры**:
  + Google Chrome (версии 90 и выше)
  + Safari (версии 12 и выше)
* **Актуальная версия программное обеспечения** JavaScript (ES6), React.js.
* **Установленная программа WebStorm**

### **3.4.2. Обоснование выбора состава технических и программных средств**

Выбор данного состава технических и программных средств обусловлен рекомендуемыми системными требованиями операционной системы Microsoft Windows 10.

# **4. Технико-экономические показатели**

## **4.1. Предполагаемая потребность**

Программа будет использоваться людьми, которым интересны головоломки, но не готовы купить физическую версию игры.

## **4.2. Ориентировочная экономическая эффективность**

В рамках данной работы расчет экономической эффективности не предусмотрен.

# **Приложение 1. Список используемой литературы**

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.301-79 Программа и методика испытаний. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.401-78 Текст программы. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.505-79 Руководство оператора. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.404-79 Пояснительная записка. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

# **Приложение 2. Описание и функциональное назначение полей, методов, свойств и классов**

Табл. 1 Описание серверной части, директории backend

|  |  |  |  |
| --- | --- | --- | --- |
| *Директория* | *Файл* | *Название* | *Описание* |
| сonfig | *db.js* | *connectDB* | *Асинхронный метод для подключения к базе данных* |
|  | *generateToken.js* | *generateToken* | *Метод для генерации jwt токена пользователю* |
| *controllers* | *lobbyControllers.js* | *createDuelLobby* | *POST запрос - Создает лобби для игры на двоих человек* |
|  | *lobbyControllers.js* | *getUserLobbies* | *GET запрос – получение всех игровых лобби для конкретного пользователя* |
|  | *lobbyControllers.js* | *createGroupLobby* | *POST запрос - Создает групповое лобби для игры на больше чем 2 человек.* |
|  | *lobbyControllers.js* | *removeFromGroupLobby* | *PUT запрос – удаляет пользователя из игрового лобби* |
|  | *lobbyControllers.js* | *checkUserFinished* | *GET запрос – возвращает bool ответ: закончил ли пользователь игру в лобби* |
|  | *lobbyControllers.js* | *checkUserStarted* | *GET запрос - возвращает bool ответ: начал ли пользователь игру в лобби* |
|  | *lobbyControllers.js* | *updateUserStartStatus* | *PUT запрос – Обновляет статус пользователя, что он начал игру в лобби.* |
|  | *lobbyControllers.js* | *updateUserFinishStatus* | *PUT запрос - Обновляет статус пользователя, что он закончил игру в лобби и устанавливает время прохождения.* |
|  | *lobbyControllers.js* | *getCompetitionTime* | *GET запрос – получение времени прохождения игры в конкретном лобби.* |
|  | *lobbyControllers.js* | *getAllUsersInLobby* | *GET запрос – получение всех пользователей, играющих в этом лобби* |
|  | *userControllers.js* | *allUsers* | *GET запрос – поиск пользователей по name и email используя regex* |
|  | *userControllers.js* | *registerUser* | *POST запрос – регистрация пользователя в бд.* |
|  | *userControllers.js* | *authUser* | *POST запрос – вход пользователя в систему* |
|  | *userControllers.js* | *updateUserScore* | *PUT запрос – изменение среднего времени прохождения и количества игр.* |
| *middlewares* | *authMiddleware.js* | *protect* | *Асинхронный обработчик, для проверки корректности jwt токена* |
|  | *errorMiddleware.js* | *notFound*  *errorHandler* | *Обработчики ошибок во время работы ПО* |
| *models* | *lobbyModel.js* | *gameLobbyModel* | *Схема БД для игрового лобби* |
|  | *userModel.js* | *userSchema* | *Схема БД для пользователя* |
| *routes* | *lobbyRoutes.js* | *Маршрутизация запросов lobby контроллерам* | |
|  | *userRoutes.js* | *Маршрутизация запросов user контроллерам* | |
| *Index.js* | *Основной файл для запуска программы* | | |

Табл. 2 Описание клиентской части – директория frontend/src

|  |  |  |  |
| --- | --- | --- | --- |
| *Директория* | *Файл* | *Название* | *Описание* |
| *components* | *Компоненты содержания страниц* | | |
| */additionalRendering* | *searchLoading.js* | *SearchLoading* | *Отображение загрузки страницы* |
| */authentication* | *Login.js* | *Login* | *Отображение блока аутентификации пользователя* |
|  | *Signup.js* | *Signup* | *Отображение блока ррегистрации пользователя* |
| */profile* | *profileModal.js* | *ProfileModal* | *Отображение блока с данными пользователя* |
| */userAvatar* | *userListItem* | *UserListItem* | *Отображение иконки пользователя с доп данными* |
| */mainComponents/gameBox* | *gameBox.js* | *GameBox* | *Блок отображения самого игрового лобби* |
|  | *singleGame.js* | *SingleGame* | *Отображение игры внутри блока GameBox* |
| */mainComponents/myLobbies* | *createNewLobby.js* | *createNewLobby* | *Механизм создание группого лобби в блоке myLobbies* |
|  | *myLobbies.js* | *MyLobbies* | *Блок с отображением всех игровых лобби для пользователя* |
| */mainComponents* | *topPanel* | *TopPanel* | *Верхня панель с названием игры, профилем и поиском пользователей* |
| *config* | *logic.js* | *getSender*  *getSenderFull* | *Получение информации о пользователи в лобби* |
| *context* | *appProvider.js* | *AppProvider* | *Контекст для получения данных в произвольных функциях frontend блока* |
| *pages* | *loginPage* | *LoginPage* | *Страница входа и регистрации* |
|  | *mainPage* | *MainPage* | *Основная страница с игровыми лобби* |
| *index.js* | *Основной файл для запуска frontend блока* | | |
| *app.js* | *Файл для маршрутизации запросов по страницам page* | | |

Табл. 3 Описание игровой части – директория frontend/src/src\_game

|  |  |  |  |
| --- | --- | --- | --- |
| *Директория* | *Файл* | *Название* | *Описание* |
| *board* | *Cell.js* | *Cell* | *Функция для работы с одной клеткой доски* |
|  | *GameBoard.js* | *GameBoard* | *Обработчик для функционирования игрового поля при заполнении доски фигурами* |
| *logic* | *rotateMatrix.js* | *rotateMatrix* | *Функция для переворота детали* |
| *parametrs* | *availableBoard.js* | *initialBoard* | *Инициализация игрового поля* |
|  | *availableShapes.js* | *initialPieces* | *Инициализация деталей игры* |
| *shapes* | *Piece.js* | *Piece* | *Обработчик взаимодействия с одной фигурой* |
|  | *PieceSelector.js* | *PieceSelector* | *Отображения набора фигур в едином компоненте* |
| *Game.js* | *Блок для расположения самой игры: доски – GameBoard, фигур – PieceSelector, кнопки начать, таймера и вывод результатов по завершении игры* | | |

# **ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Изм. | Номера листов | | | | Всего листов (страниц) в документе | № документа | Входя- щий № сопро- водит. докум. и дата | Под- пись | Дата |
| изменен- ных | заменен- ных | новых | аннулиро- ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |