На основе моей идеи создать небольшую браузерную игру, мною был выбран протокол HTTP для взаимодействия Сервер-Клиент. А реализовать его работу я решил через REST и вот почему:

1. **Простота использования и понимания**: REST предоставляет простой и интуитивно понятный способ организации взаимодействия между клиентом и сервером с использованием стандартных HTTP методов, таких как GET, POST, PUT и DELETE. Это делает его отличным выбором для проектов, где важна ясность интерфейса и простота взаимодействия.
2. **Широкая поддержка и распространенность**: REST является одним из самых распространенных подходов к построению веб-сервисов и API. Он широко поддерживается в различных языках программирования и фреймворках, что обеспечивает легкость разработки и интеграции.
3. **Гибкость и масштабируемость**: REST не накладывает строгих ограничений на структуру данных или формат сообщений, позволяя разработчикам проектировать API с учетом конкретных требований проекта. Кроме того, благодаря своей распределенной природе и использованию стандартных протоколов HTTP, REST API легко масштабируются и могут быть расширены по мере необходимости.
4. **Поддержка различных форматов данных**: REST позволяет работать с различными форматами представления данных, такими как JSON, XML, HTML и другими. Это обеспечивает мне гибкость в выборе формата данных в зависимости от моих предпочтений и текущих задач.
5. **Совместимость с существующей инфраструктурой**: Поскольку ваш сервер уже использует HTTP протокол для выдачи web страниц клиентам, выбор REST позволит мне легко интегрировать новые функции с существующей инфраструктурой без дополнительных сложностей.

| **API** | **Описание** | **Пример** |
| --- | --- | --- |
| **menu** | Отображает страницу меню. | GET / |
| **game** | Отображает страницу игры. | GET /game |
| **docs** | Отображает страницу документации. | GET /docs |
| **waitroom** | Обрабатывает регистрацию игроков и начало игры. | POST /wait\_room { Player: "Me", Use\_AI: True } |
| **clearplayer** | Удаляет игрока из списка игроков. | DELETE /wait\_room { Player: "Me" } |
| **send\_grid** | Отправляет текущее состояние игровой сетки клиенту в формате JSON. | GET /send\_grid |
| **turn** | Обрабатывает ход игрока, обновляя игровую сетку выбранным цветом.  Если ходит ИИ, ИИ-игрок делает ход автоматически. | POST /turn { Player: "Me", Color: "#FF0000" } |

Реализацию моего решения можно увидеть в таблице (Таблица 1):

Таблица 1 – Документация API Доступная по запросу GET /docs