Архитектура взаимодействия фронтенда и бэкенда в «Click & Cook»

1. Разделение логики между фронтендом и бэкендом

Фронтенд (Angular) – Клиентская часть

Что обрабатывается на фронте:

- Интерфейс пользователя (UI/UX).
- Клики игрока (увеличение денег и количества блюд в локальном состоянии).
- Визуализация изменений (анимации, таймеры, улучшения).
- Локальный кэш краткосрочных данных (например, последние секунды прогресса).
- Запросы на сервер для сохранения и загрузки данных.
- Управление временными бустами и эффектами в реальном времени.
- Обновление клиентского состояния на основе ответов сервера.

Бэкенд (Go) – Серверная часть

Что обрабатывается на бэке:

- Генерация уникальных игровых сессий для пользователей.
- Хранение и валидация данных (прогресс, улучшения, экономика).
- Верификация кликов и автоматических доходов для предотвращения читерства.
- Управление престижем и начислением бонусов.
- Обработка событий (бонусы).
- Взаимодействие с базой данных.
- Управление пользовательскими сессиями и авторизацией.

2. Взаимодействие между фронтендом и бэкендом

Запросы от клиента к серверу

Общение между фронтендом и бэкендом можно реализовать через:

- **REST API** – для периодического обновления данных (получение прогресса, сохранение улучшений).
- **WebSockets** – для передачи в реальном времени изменений (пассивный доход, временные бонусы).

3. Пример работы сессии игрока

1. Игрок заходит на сайт → Фронтенд отправляет запрос `GET /game/init`.
2. Сервер проверяет, есть ли сохраненная игра:
    - Если **да** → Возвращает прогресс игрока.
    - Если **нет** → Создает новую сессию.
3. Игрок начинает кликать по блюду → Фронт отправляет `PATCH /game/cook`, а сервер обновляет блюда.
4. Игрок начинает кликать по продаже → фронт отправляет `PATCH /game/sell`, а сервер обновляет баланс.
5. Игрок покупает улучшение → Фронтенд обновляет UI и отправляет `PATCH /game/buy/:ID`.
6. Сервер проверяет возможность покупки, списывает деньги и обновляет доход.
7. Пассивный доход и события работают через WebSockets:
    - Сервер раз в 3 секунды отправляет клиенту новые начисления денег.
    - Если начинается событие, сервер рассылает уведомление игрокам.
8. Если игрок выходит и заходит позже → Сервер подгружает данные из БД.

4. Оптимизация и предотвращение читерства

Чтобы предотвратить накрутку денег, сервер должен проверять данные:

- **Антиспам кликов** – Если игрок отправляет запросы слишком часто, сервер может их заблокировать.
- **Серверная валидация улучшений** – Перед покупкой улучшения сервер проверяет баланс игрока.

Вывод

Таким образом, кликер можно реализовать следующим образом:

1. Фронт (Angular) отвечает за UI, анимации, локальное хранение кликов и обновлений.
2. Бэк (Go) управляет бизнес-логикой, сохраняет данные, обрабатывает запросы и отправляет обновления через WebSockets.
3. REST API используется для аутентификации, сохранения и загрузки данных.
4. WebSockets применяются для динамических обновлений и пассивного дохода.
5. База данных хранит прогресс игроков, улучшения, события и статистику.