Documentation

Decentralized NFT Card Game

СОДЕРЖАНИЕ

1 Функциональные требования ... 3
1.1 Подключение кошелька ... 3
1.2 Покупка NFT-карт ... 3
1.3 Управление инвентарем ... 3
1.4 Создание матча ... 4
1.5 Присоединение к матчу ... 4
1.6 Разрешение матча ... 4
1.7 Торговля ... 4
1.8 История матчей ... 5
1.9 Диаграмма вариантов использования ... 5
2 Технические требования ... 6
2.1 Основная сеть ... 6
2.2 Клиент ... 6
2.3 Сервер ... 7
3 Пользовательский интерфейс ... 11
3.1 Общие сведения ... 11
3.2 Экран загрузки ... 11
3.3 Домашний экран ... 12
3.4 Магазин ... 12
3.5 Лобби ... 13
3.6 Экран матча ... 14
3.7 Экран инвентаря ... 14
3.8 Экран профиля ... 14
3.9 История матчей ... 15
3.10 Настройки ... 15
4 Сценарии тестирования ... 16

1 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Функциональные требования к приложению *DNCG* определяют его поведение, обеспечивая, чтобы каждый аспект игры – от подключения пользователя до разрешения матчей и управления активами – работал предсказуемо, безопасно и интуитивно, с полным учётом децентрализованной природы на блокчейне *Solana*. Эти требования охватывают весь пользовательский путь, начиная с момента входа в систему и заканчивая долгосрочным взаимодействием, где акцент сделан на прозрачности, атомарности транзакций и интеграции с крипто-кошельками, чтобы игрок чувствовал себя в полной безопасности и контроле над своими активами.

1.1 Подключение кошелька

Сначала система должна позволять пользователю беспрепятственно подключать совместимый *Solana*-кошелёк, такой как *Phantom* или *Solflare*, через стандартный адаптер, где после подтверждения в кошельке приложение автоматически загружает публичный ключ, баланс в *SOL* и *USDC*, а также список принадлежащих токенов $*DNCG*. Это подключение сохраняется в локальном хранилище браузера для последующих сессий, но с возможностью отключения в любой момент, и сразу после авторизации отображается персонализированный дашборд с обзором инвентаря, статистики матчей и доступных действий, чтобы новичок мог сразу ориентироваться без лишних шагов.

1.2 Покупка NFT-карт

Далее следует механизм покупки *NFT*-карт, где пользователь переходит в магазин, видит три варианта – Камень, Ножницы или Бумага – с детальными превью, включая анимированные изображения и метаданные, хранящиеся на *Arweave*. Выбор типа карты приводит к формированию транзакции на 1 SOL, которая подтверждается в кошельке, после чего смарт-контракт через *Metaplex* минтит уникальный *NFT*-токен, привязанный исключительно к адресу покупателя, с автоматическим обновлением инвентаря в реальном времени. Если средств недостаточно или транзакция прерывается, система обрабатывает ошибку, возвращая пользователя к выбору с информативным сообщением, и логирует событие для аналитики.

1.3 Управление инвентарем

Просмотр и управление инвентарём реализованы как центральный хаб, где все *NFT*-карты отображаются в гриде с фильтрами по типу, редкости и дате приобретения, с возможностью кликнуть на карту для детального вида – включая ссылку на *Solana Explorer* для верификации *ownership* и метаданных. Отсюда пользователь может инициировать продажу на внешних

маркетплейсах через встроенные кнопки, которые генерируют прямые ссылки на *Magic Eden* или *Tensor*, или использовать карту для создания матча, где система проверяет, что карта не заблокирована в другом эскроу.

1.4 Создание матча

Создание матча начинается с выбора карты из инвентаря, после чего приложение формирует инструкцию для смарт-контракта, блокирующую *NFT* в *PDA*-эскроу и регистрирующую матч как открытый с уникальным *ID*, видимым в лобби. Пользователь видит подтверждение блокировки и может поделиться ссылкой на матч через социальные сети, а система в реальном времени обновляет список доступных игр для других игроков, с фильтрами по ставке или типу карт, чтобы облегчить поиск подходящего оппонента.

1.5 Присоединение к матчу

Присоединение к матчу происходит аналогично: пользователь видит детали открытого матча, выбирает свою карту, подтверждает транзакцию, и после успешной блокировки второй *NFT* в эскроу матч автоматически переходит в активный статус, с уведомлением обоим игрокам через браузерные нотификации или *WebSocket*. Здесь критично, что система предотвращает присоединение с несовместимой картой или от того же адреса, возвращая ошибку с объяснением.

1.6 Разрешение матча

Разрешение матча – это кульминация, где смарт-контракт автономно сравнивает типы карт по фиксированным правилам, определяет победителя или ничью, и в одной атомарной транзакции переводит обе *NFT* на кошелёк победителя, возвращая их при ничьей, с удержанием 1 процента комиссии в *SOL* или со скидкой при использовании $*DNCG*. Результат мгновенно отражается в *UI* с анимацией, обновлением статистики и записью в историю, где каждый матч имеет *permalink* для просмотра в Explorer, включая хэши транзакций и использованные случайные числа от *Pyth*, если они задействованы.

1.7 Торговля

Торговля и монетизация встроены *seamlessly*: рядом с каждой картой кнопка для выставления на продажу, которая открывает модальное окно с предложением цены и прямой интеграцией с маркетплейсами, а для $*DNCG* – своп через *Raydium* прямо в приложении.

1.8 История матчей

История матчей и статистика предоставляют глубокую аналитику: пользователь видит хронологический лог с фильтрами по исходу, оппонентам и датам, с графиками побед/поражений, заработанных *NFT* и потраченных комиссий, где каждая запись кликабельна для детального разбора транзакций.

1.9 Диаграмма вариантов использования

На рисунке 1.1 представлена диаграмма вариантов использования веб-приложения.

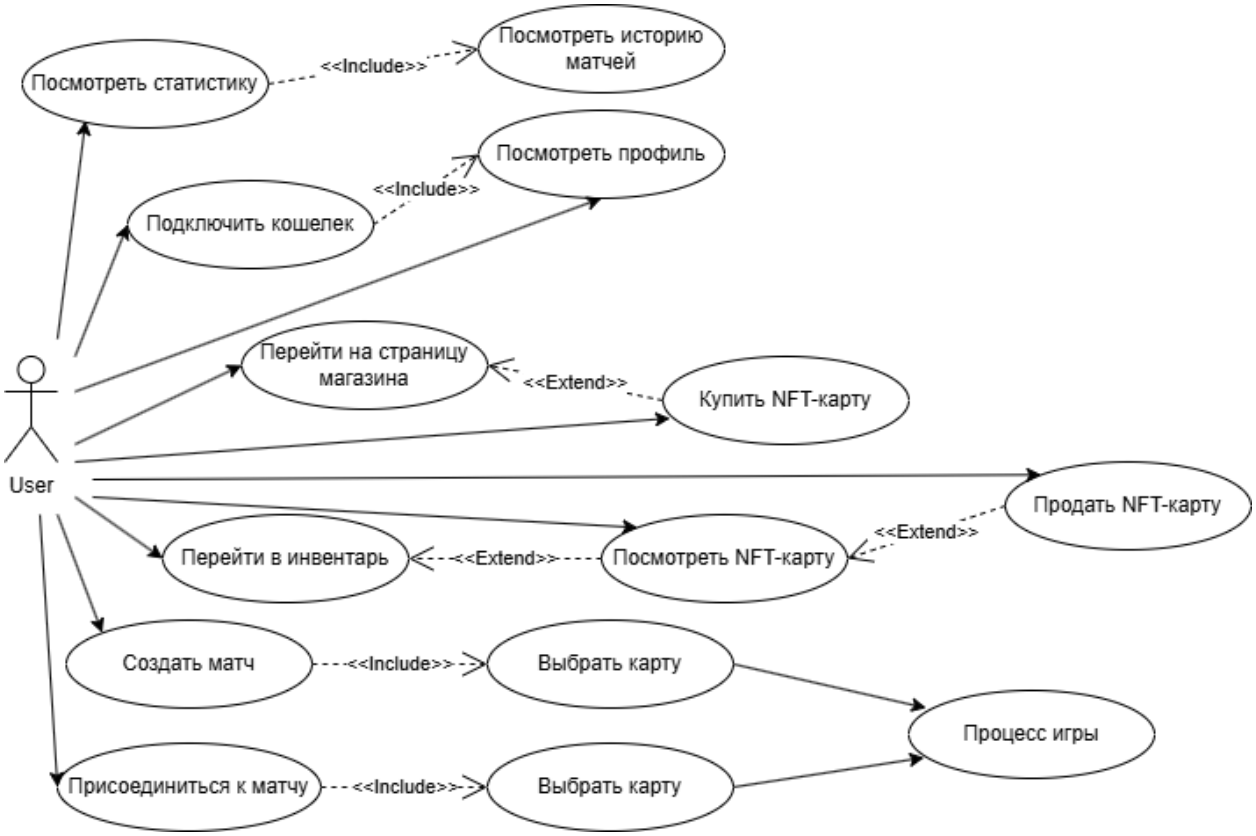

Рисунок 1.1 – Usecase диаграмма

2 ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Технические требования к проекту *DNCG* формируют надёжный фундамент для создания высокопроизводительной, безопасной и полностью децентрализованной игровой платформы на блокчейне *Solana*, где каждый компонент оптимизирован для минимальных задержек, низких комиссий и максимальной устойчивости к атакам, обеспечивая непрерывный опыт для тысячи пользователей.

2.1 Основная сеть

В основе лежит выбор *Solana* как основной сети, с использованием *Devnet* для разработки и тестирования, где транзакции обрабатываются с скоростью более двух тысяч в секунду при комиссиях в доли цента, и последующим переходом на *Mainnet* для продакшена, где все взаимодействия верифицируются через *RPC*-ноды от надёжных провайдеров вроде *Helius* или *QuickNode*, гарантируя доступность даже в пиковые нагрузки.

Смарт-контракты разрабатываются на языке *Rust* версии 1.70 или выше с фреймворком *Anchor* версии 0.30 и новее, который автоматизирует обработку аккаунтов, инструкций и ошибок, минимизируя риски человеческих ошибок, при этом интеграция с *Metaplex Token Metadata* обеспечивает стандартный минтинг *NFT* с перманентным хранением метаданных на *Arweave* через *Bundlr*, где изображения карт, анимации и атрибуты остаются неизменными и доступными навсегда без зависимости от централизованных серверов.

Для обеспечения проверяемой честности в разрешении матчей используется оракул *Pyth Network*, предоставляющий криптографически защищённые случайные числа в реальном времени, с *fallback* на *Switchboard* в случае сбоев, а все эскроу-операции реализуются через *Program Derived Addresses*, где *NFT* блокируются исключительно под контролем программы, исключая возможность кражи или манипуляции. Токен $*DNCG* создаётся как *SPL*-токен с фиксированной эмиссией, с поддержкой стейкинга и свопа через *Raydium*, где ликвидность пополняется автоматически из комиссий.

2.2 Клиент

Клиентская часть – это всё, что работает в браузере пользователя, построена на *React.js* с *TypeScript* и полностью *stateless*, где состояние игры синхронизируется напрямую с блокчейном через *RPC*-запросы и *WebSocket*. Пользователь подключает кошелёк (*Phantom* или аналог), и весь *UI* – от дашборда с инвентарём до лобби матчей – рендерится локально, запрашивая данные через библиотеки @solana/web3.js и @project-serum/anchor, которые генерируют и подписывают транзакции на стороне клиента. Например, покупка *NFT*-карты формирует инструкцию *mint_card*, отправляемую прямо на *Solana RPC*, без промежуточных серверов, с мгновенным обновлением *UI*

через подписки на события программы. Реал-тайм уведомления о матчах реализуются через *WebSocket* к *RPC*-ноде, где клиент слушает изменения в *PDA*-эскроу или аккаунтах матчей, отображая анимации и результаты без перезагрузки страницы. Локальное хранение ограничено *localStorage* для предпочтений (тема, язык) и *sessionStorage* для временного кэша публичных ключей, без приватных данных.

2.3 Сервер

Серверная часть – опциональная и минимальная, реализована как лёгкий *Node.js API* на *Express.js*, хостимый на *Vercel* или *AWS Lambda* для *serverless* масштабирования, где она выступает только как индексатор и прокси для оффчейн-данных, не храня приватные ключи или критическую логику. Сервер подключается к *Solana* через *RPC* и *WebSocket*, периодически сканируя программу с *getProgramAccounts* для кэширования активных матчей, истории или статистики, чтобы ускорить загрузку лобби для клиентов – вместо полного сканирования цепи браузером. Для уведомлений сервер использует *push*-сервисы (*Web Push API*) или интеграцию с *Telegram*/*Discord* ботами, отправляя алерты о завершённых матчах на основе событий. Аналитика (популярные карты, топ-игроки) агрегируется здесь и сервируется через *REST*/*GraphQL* эндпоинты, с кэшированием в *Redis* для скорости.

База данных – *PostgreSQL* или *MongoDB* для хранения оффчейн-метаданных, таких как пользовательские ники, кастомные аватары или кэш событий (не транзакции, а агрегированные данные вроде «матч *ID X* завершён»), с шифрованием. Сервер не подписывает транзакции – только читает и проксирует, с аутентификацией через *JWT* от кошелька. Это добавляет удобство: быстрее поиск матчей, персонализированные рекомендации, но остаётся опциональным – клиент может работать на прямых *RPC*-запросах. На рисунке 2.1. представлены основные таблицы.

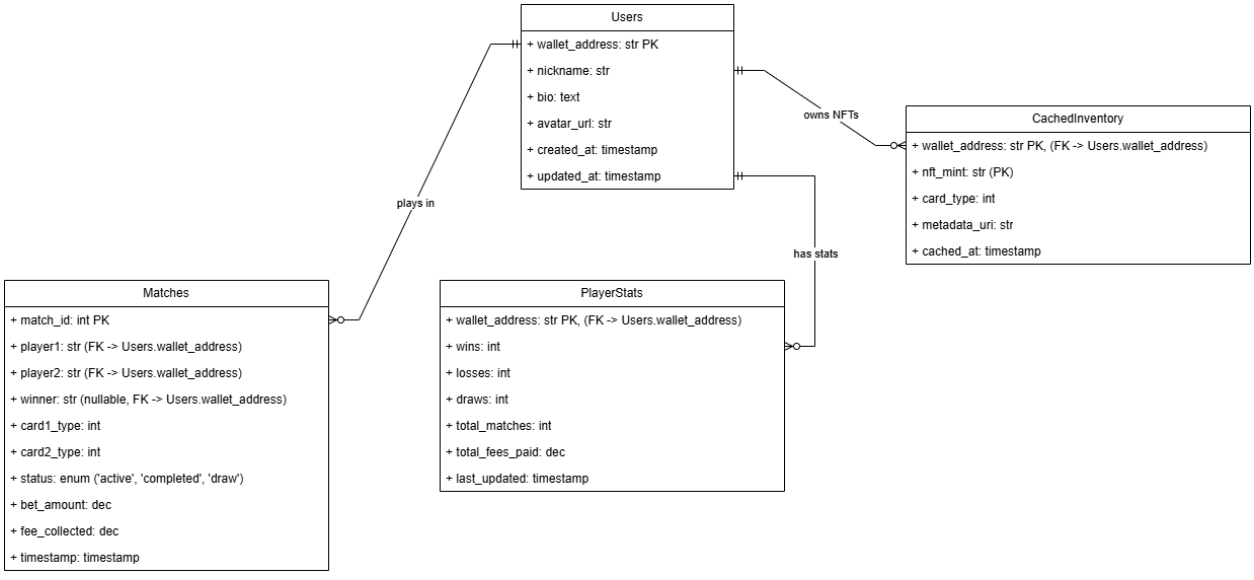

Рисунок 2.1 – Таблицы базы данных

В приложении *DNCG* все эти функции – статистика, история матчей, профиль, изменения профиля и инвентарь – полностью реализуемы в децентрализованной архитектуре на *Solana*, где основной акцент на *on-chain* данных для прозрачности и безопасности, с минимальным использованием оффчейн-компонентов для удобства и скорости. Это сохраняет принципы *Web3*: пользователь владеет своими данными через кошелёк, а приложение не хранит ничего централизовано, кроме кэша для производительности.

Инвентарь реализован как центральный компонент дашборда, где клиент напрямую запрашивает список *NFT*-карт, принадлежащих подключённому кошельку, через вызовы к *Metaplex* и *Solana RPC*: библиотека @solana/web3.js получает все токен-аккаунты по владельцу, фильтрует по *mint*-адресам программы *DNCG*, и загружает метаданные с *Arweave* для отображения изображений, типов (Камень/Ножницы/Бумага) и редкости.

Серверная часть здесь опциональна: лёгкий индексатор на *Node.js* кэширует инвентарь в *Redis* по публичному ключу, обновляя каждые 30 секунд, чтобы ускорить загрузку для мобильных пользователей, но клиент всегда может *fallback* на прямой *RPC*-запрос. Изменения инвентаря (покупка или выигрыш) триггерят *WebSocket*-уведомление, автоматически обновляя *UI* без перезагрузки. Оффчейн БД (*MongoDB*) хранит только кастомные метки, если пользователь добавит заметки к картам, но это локально в браузере через IndexedDB.

История матчей строится на *on-chain* событиях: каждый матч оставляет неизменяемую запись в *PDA*-аккаунтах программы, с полями *ID*, игроками, типами карт, исходом и *timestamp*. Клиентская часть собирает историю через *getProgramAccounts* с фильтрами по владельцу, парсит логи транзакций и отображает в хронологическом списке с пагинацией, где каждая запись расширяется до детального вида с анимацией, хэшами и кнопкой «Поделиться». Для реал-тайм сервер индексирует события через *WebSocket*-подписку на программу, сохраняя агрегированные данные в *PostgreSQL* (матч *ID*, *winner*, *fee*), и предоставляет *API* /*api*/*history*/{*wallet*} для быстрого *JSON*, с кэшем в *Redis* на 5 минут. Это ускоряет загрузку для длинной истории (сотни матчей), но клиент может игнорировать сервер и сканировать цепь напрямую. Уведомления о новых записях пушатся через *Web Push API* от сервера, основываясь на подписке пользователя.

Статистика агрегируется динамически: клиент рассчитывает базовые метрики (победы/поражения, *winrate*, заработанные *NFT*) локально из истории, используя *lodash* для обработки массивов, с графиками на *Chart.js* для визуализации. Более сложные *insights* – топ оппонентов или глобальный рейтинг – приходят от сервера, который сканирует все матчи, обновляя таблицы в БД с индексами для быстрых запросов, и сервирует /*api*/*stats*/{*wallet*} или глобальный /*api*/*leaderboard*. Это оффчейн для скорости, но верифицируемо: пользователь может кликнуть «Проверить *on-chain*» для перерасчёта.

Профиль пользователя привязан к кошельку, без логинов: клиент отображает *pubkey*, баланс *SOL*/$*DNCG* и аватар (из *NFT* или *Gravatar* по

адресу), со статистикой в табах. Изменения профиля – это оффчейн возможность: пользователь редактирует ник, био или аватар в форме, которая сохраняет данные в *IndexedDB* локально и синхронизирует с сервером через *POST /api/profile*, где они хранятся в *MongoDB* по хэшу *pubkey* (без приватных данных). Сервер валидирует подпись сообщения от кошелька для аутентификации, предотвращая спам, и пушит обновления через *WebSocket*. Если сервер недоступен, изменения остаются локальными. Для глобального профиля (виден оппонентам) используется *on-chain PDA* с опциональными метаданными, минтенными как *NFT*. На рисунке 2.2 представлена диаграмма классов.

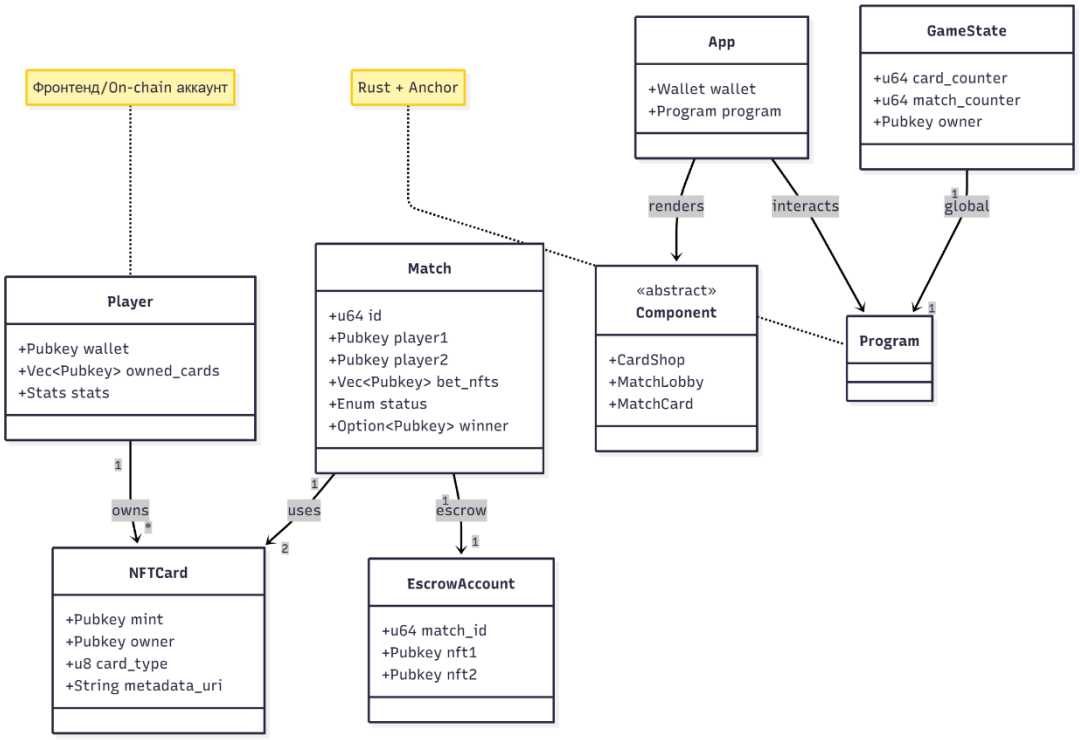

Рисунок 2.2 – Диаграмма классов

Взаимодействие между частями происходит через публичные *API*: клиент вызывает сервер для читаемых данных (*GET*), но для записи (транзакции) – напрямую к *Solana*, минимизируя доверие. БД не хранит чувствительные данные – только публичные события, с возможностью миграции на полностью децентрализованные решения в будущем. Это разделение делает *DNCG* гибким: в *MVP* сервер можно отключить, полагаясь только на клиент и цепь, а позже масштабировать для миллионов пользователей без компромиссов в децентрализации. На рисунке 2.3 представлена диаграмма последовательностей для матча.

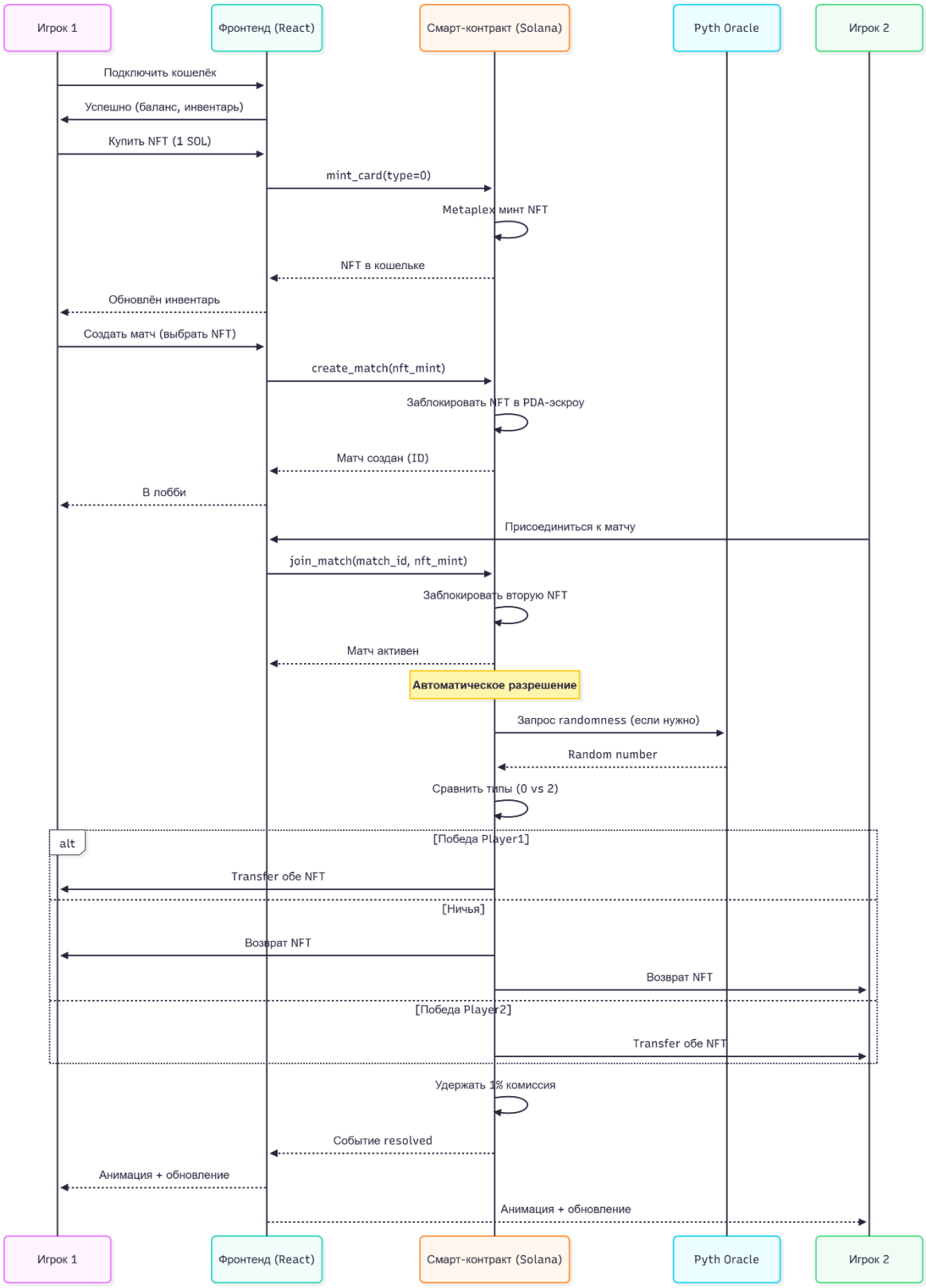

Рисунок 2.3 – Диаграмма последовательностей

3 ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

3.1 Общие сведения

Интерфейс приложения *DNCG* представляет собой современный, минималистичный, оптимизированный для браузеров на десктопах и мобильных устройствах, с тёмной темой по умолчанию (переключаемой на светлую), акцентами в неоново-зелёном и фиолетовом цветах *Solana*, и анимациями для плавных переходов между экранами. Всё построено на *React* с *React Router* для навигации, где основной *layout* включает фиксированный хедер с логотипом *DNCG*, кнопкой подключения кошелька (*Phantom*/*Solflare*), балансом *SOL*/$*DNCG* и меню (*Dashboard*, *Shop*, *Lobby*, *Profile*, *History*), плюс футер с ссылками на социальные сети. Экраны переключаются без релоада, с ленивой загрузкой компонентов для скорости. Ниже подробное описание каждого экрана, как он выглядит, что на нём происходит и как пользователь взаимодействует, с акцентом на интуитивность даже для новичков в *Web*3.

3.2 Экран загрузки

Экран загрузки представлен на рисунке 3.1.

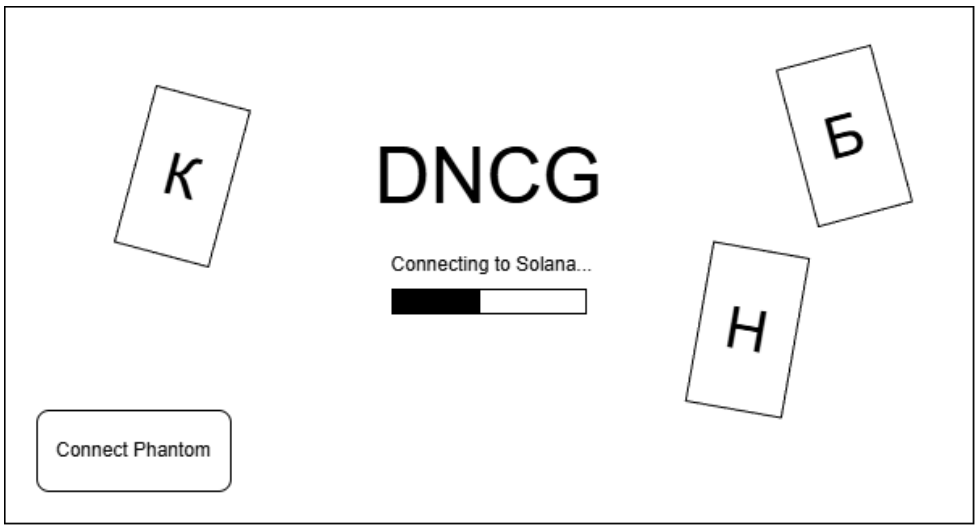

Рисунок 3.1 – Экран загрузки

Экран загрузки (*Splash Screen*) встречает пользователя при первом открытии: полноэкранный фон с градиентом от тёмно-синего к чёрному, центрированный логотип *DNCG* в виде стилизованных карт (Камень-Ножницы-Бумага в неоновом свечении), анимация вращения и текст «*Connecting to Solana*...» с прогресс-баром. Если кошелёк не подключён, появляется модальное окно с кнопками «*Connect Wallet*» и списками поддерживаемых (Phantom, Solflare), плюс QR-код для мобильных. После

подключения или если сессия сохранена, идет плавный переход на экран *Dashboard* за 1 секунду, с сохранением состояния в *localStorage*.

3.3 Домашний экран

Домашний экран представлен на рисунке 3.2.

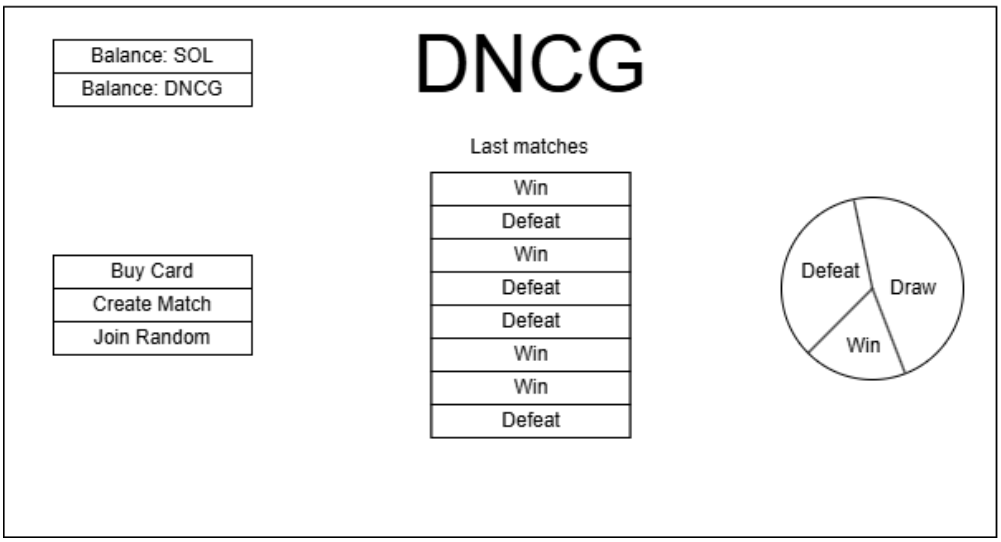

Рисунок 3.2 – Домашний экран

Dashboard – это домашний экран после входа, разделённый на три колонки на десктопе и стек на мобильных: левая – быстрые действия (кнопки «*Buy Card*», «*Create Match*», «*Join Random*»), центральная – карусель с последними матчами, правая – мини-статистика (*Wins*/*Losses*/*Draws* в круговых диаграммах на *Chart.js*, с winrate в большом неоновом числе). Вверху – приветствие «Welcome back» (ник или укороченный pubkey), баланс в карточках *SOL* и $*DNCG*. Ниже может быть лента новостей: «*New rare card dropped*!» или «*You won* 2 *NFT*!» с уведомлениями, которые можно свайпнуть. Фон с лёгкой анимацией частиц, реагирующих на курсор.

3.4 Магазин

Shop (Магазин) – экран покупок *NFT*, с секцией «*Mint Your Card*» и тремя большими карточками: Камень (скала с трещинами), Ножницы (блестящие лезвия), Бумага (лист с ветром), каждая с *hover*-анимацией (увеличение, свечение) и деталями (цена 1 SOL, редкость *Common*, кнопка «*Mint Now*»). После клика – модальное подтверждение с превью *NFT* и кнопкой «*Confirm in Wallet*», где после успеха карта добавляется в инвентарь с тост-уведомлением «*Minted*! *Check your inventory*». Внизу – галерея редких карт (будущие дропы) и *FAQ* «*How minting works on Solana*».

Экран магазина представлен на рисунке 3.3.

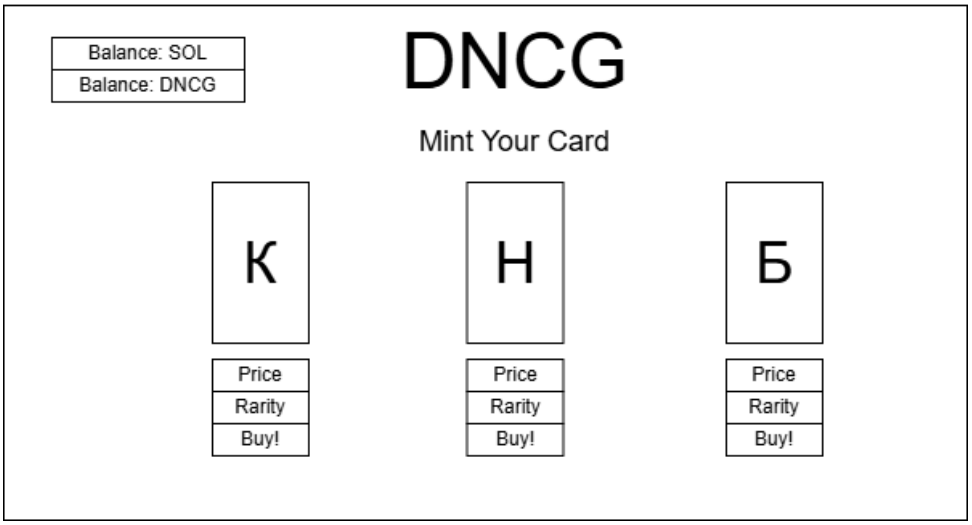

Рисунок 3.3 – Экран магазина

3.5 Лобби

Экран лобби представлен на рисунке 3.4.

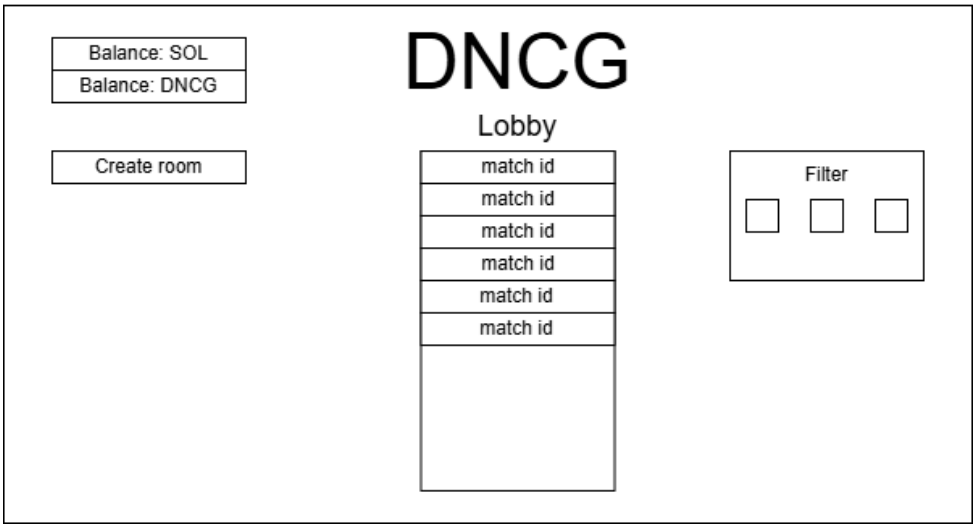

Рисунок 3.4 – Экран лобби

Lobby (Лобби матчей) – динамичный экран с поиском и фильтрами: поиск по *ID* матча, фильтры, сортировка по времени. Центральный список – карточки активных матчей в виде таблиц или грида: аватарки игроков (из профиля или дефолтные), типы карт (скрытые до присоединения), ставка «2 NFT», таймер «Waiting for opponent» с пульсирующей анимацией. Кнопка «Create Match» открывает модал с выбором вашей карты из инвентаря, после – матч появляется в топе. Для присоединения – клик на матч, превью оппонента, выбор вашей карты, подтверждение в кошельке.

3.6 Экран матча

Экран матча представлен на рисунке 3.5.

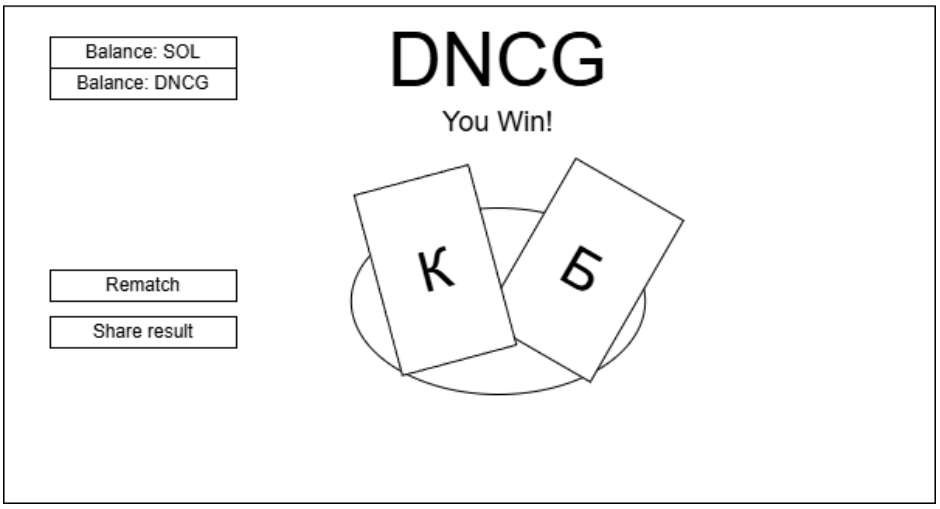

Рисунок 3.5 – Экран матча

Match View (Экран матча) – отдельный роут для активного или завершённого матча, с полноэкранной анимацией: две карты вылетают с боков, сталкиваются в центре с эффектами, текст «You Win!» или «Draw» в огромном шрифте. Ниже – детали: карты игроков с метаданными, *tx_signature* ссылкой на *Explorer*, кнопки «*Rematch*» или «*Share result*».

3.7 Экран инвентаря

Inventory (Инвентарь) – экран коллекции, грид из 3-6 карточек на строку, каждая с изображением, типом, редкостью (*Common*/*Rare* с бейджами), датой приобретения и кнопками «*Use in Match*», «*Sell*» (открывает внешнюю ссылку) или «*View on Explorer*». Фильтры сверху: по типу, редкости, сортировка. Hover – 3D-поворот карты с бэк-сайдом (статистика использования). Пустой инвентарь – иллюстрация с призывом «*Mint your first card*!» и кнопкой в *Shop*.

3.8 Экран профиля

Profile (Профиль) – персональная страница с аватаром (круглый, кликабельный для смены), ником (редактируемым), био, статистикой в карточках (*Wins*/*Losses*/*Draws*, *Total Matches*, *Winrate*, *Total Fees Saved* с $*DNCG*), и графиком активности (линейный чарт по дням). Ниже – достижения (бейджи «*First Win*», «10 *Draws Streak*») и кнопка «*Edit Profile*» для модала с формой (ник, био, аватар *upload*). Публичный профиль (по *wallet*) виден оппонентам в матчах: мини-версия с *winrate* и последними матчами.

Экран профиля представлен на рисунке 3.6.

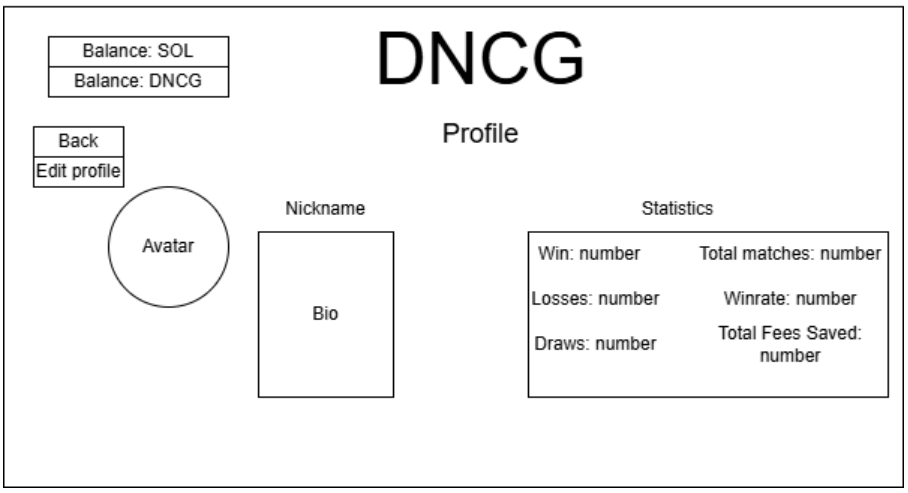

Рисунок 3.6 – Экран профиля

3.9 История матчей

History (История матчей) – бесконечный скролл списка с пагинацией, каждая запись – карточка с датой, оппонентами (аватарки), вашим выбором *vs* их, исходом (зелёный/красный/серый), выигранными *NFT* и кнопкой «*Details*» для *Match View*. Фильтры: по исходу, дате, оппоненту. Топ – поиск по *ID*. Экспорт в *CSV* кнопкой «*Download History*».

3.10 Настройки

Settings (Настройки) – доступны из хедера: переключатель темы, язык (*EN*/*RU*), уведомления (*push* для матчей), подключённые кошельки (*disconnect*), и «*On-chain only mode*» для отключения сервера. Плюс ссылки на *Whitepaper*, *GitHub* и *Support*.

Модальные окна унифицированы: тёмный оверлей, центрированный контент с крестиком, анимация *scale-in*. Уведомления – тосты в правом нижнем углу с таймаутом. Всё адаптивно: на мобильных хедер сворачивается в гамбургер, гриды становятся колонками, кнопки большими для тача. Это делает интерфейс не просто функциональным, а увлекательным – как игра внутри игры, где каждый экран мотивирует играть дальше.

4 СЦЕНАРИИ ТЕСТИРОВАНИЯ

Тестирование является ключевым этапом разработки *DNCG*, обеспечивая надёжность, безопасность и прозрачность децентрализованной игровой платформы на *Solana*. Мы применяем многоуровневый подход: от изолированных *unit*-тестов смарт-контрактов до комплексных *end-to-end* сценариев в реальном браузере, с акцентом на атомарность транзакций, защиту от уязвимостей и устойчивость к нагрузкам. Все тесты автоматизированы, интегрированы в *CI*/*CD* (*GitHub Actions*) и используют инструменты вроде *Anchor* для *Rust*, *web*3*.js* для интеграции и *Cypress* для *UI*. Покрытие кода превышает 90%, с обязательным внешним аудитом перед *mainnet*. Это гарантирует, что игра работает честно, без простоев и манипуляций, даже при тысячах одновременных матчей.

Далее будут описаны сценарии тестирования для смарт-контрактов (*Rust*/*Anchor*):

1 Тестирование минтинга *NFT*. Создать тестовый кошелёк, пополнить *SOL*, вызвать *mint_card* с типом карты (0-2), проверить создание токен-аккаунта в *Metaplex*, *ownership* на покупателе, метаданные на *Arweave* и инкремент счётчика в *GameState*.

2 Тестирование логики матча. Мок *Pyth* для *randomness*, заблокировать две *NFT* в *PDA*-эскроу, вызвать *resolve_match*, проверить сравнение типов (победа/поражение/ничья), *transfer NFT* победителю, удержание 1% комиссии (или 0.8% с $*DNCG*), эмит событий.

3 *Edge*-кейсы. Недостаток *SOL* (ошибка), неверный тип карты (*assert_err*).

Далее будут описаны *Integration*-тесты (*Devnet*, *web*3*.js*):

1 Полный матч с двумя кошельками: минт карт, *create_match*, *join_match*, *resolve*, проверить *transfer NFT*, комиссию в казне, обновление истории *on-chain*.

2 Параллельные матчи. 10 одновременных, проверить уникальность *PDA*, отсутствие конфликтов аккаунтов.

Далее будут описаны *End-to-End* тесты (*Cypress*, браузер с *Phantom*):

1 Пользовательский поток. Подключить кошелёк, купить карту, создать матч, второй браузер присоединяется, анимация разрешения, обновление инвентаря/истории/статистики.

2 Профиль. Редактирование ника/био, сохранение в *IndexedDB* + сервер, верификация подписи.

3 Уведомления. Пуш при завершении матча, *WebSocket* обновления лобби.

Далее будут описаны стресс-тесты (*custom script*):

1 1000 одновременных матчей. Генерация ключей, параллельные промисы, мониторинг *TPS*, задержек (<2s), памяти *RPC*.

2 Фронтенд. 500 *WebSocket*-соединений, проверка *UI* на лаги (*React Profiler*).

Далее будут описаны *Edge*-кейсы и регрессия:

1 Ничья в 100% матчах. Возврат *NFT*, нулевая комиссия.

2 Оффлайн/сеть. *PWA* кэш истории, *graceful* ошибки.

3 Смена кошелька. Сброс сессии, миграция данных.

4 Пост-апдейт. *Upgrade* программы, проверка старых аккаунтов.

В заключение, эти сценарии тестирования превращают *DNCG* в доверенную платформу, где каждый матч – это не только азарт, но и гарантия честности благодаря блокчейну. Регулярные тесты и открытый код позволяют сообществу участвовать в верификации, обеспечивая эволюцию проекта без компромиссов в безопасности или производительности.