Факультет Компьютерных Наук

Программная инженерия

Москва 2025

Space Hunter: Telegram игра с NFT и встроенной валютой

Выполнили:
Дедов Иван Андреевич, студент группы БПИ-234
Гетманова Карина Сергеевна, студентка группы БПИ-234
Крючков Сергей Алексеевич, студент группы БПИ-234

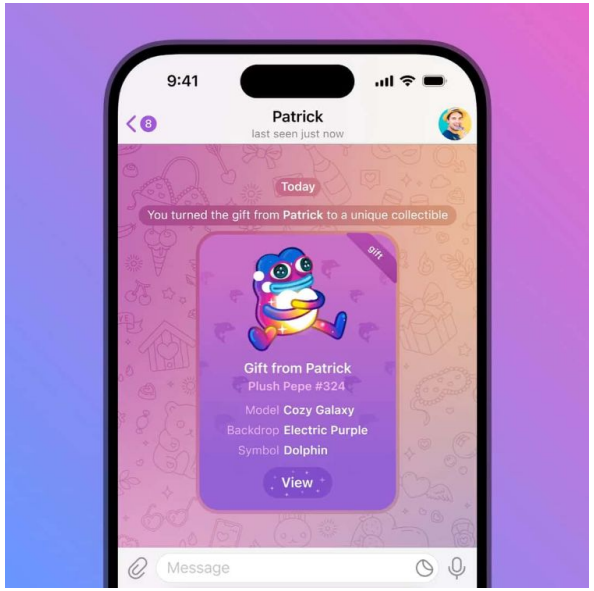
Научный руководитель проекта:
Штатный преподаватель департамента больших данных и информационного поиска:
Янович Юрий Александрович

Описание предметной области

Space Hunter — это веб-приложение, написанное для формата Telegram Mini Apps, объединяющее элементы кликера и коллекционной игры с интеграцией NFT-технологий и криптовалютного кошелька TON. Игра позволяет пользователям исследовать космическое пространство, коллекционировать уникальные NFT объекты космической тематики и выполнять задания для получения бонусных предметов.

Актуальность работы

Современные пользователи мессенджеров проявляют повышенный интерес к микро развлечениям внутри привычных платформ связи. Интеграция игровых механик с блокчейн-технологиями отвечает современным тенденциям геймификации и монетизации цифрового контента, а космическая тематика остается одной из наиболее привлекательных и универсальных для широкой аудитории. Приложение позволяет познакомить пользователей с современными технологиями через привычный интерфейс Telegram.



Цель и задачи

Цель работы: Разработать интерактивное Telegram-приложение, совмещающее игровые механики с блокчейн-технологиями для вовлечения пользователей в экосистему Web3.

Задачи:

1. Создать увлекательную игровую механику на основе кликера с системой прогрессии
2. Разработать коллекционную систему NFT с тематической классификацией
3. Внедрить систему заданий и достижений для стимулирования игрового прогресса
4. Реализовать интеграцию с блокчейн TON для хранения цифровых активов
5. Обеспечить высокую производительность и отзывчивость приложения внутри платформы Telegram

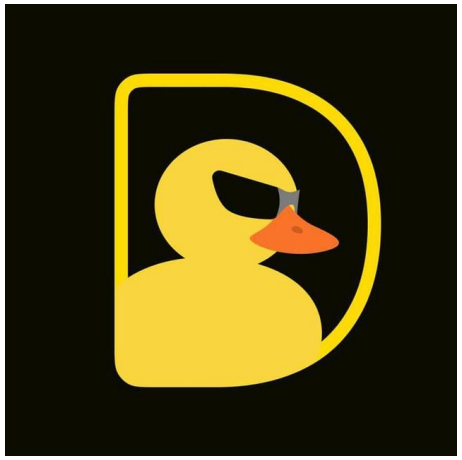
Сравнение-анализ аналогов и существующих решений

@hamster_kombat_bot

@tapswap_bot

@theYescoin_bot

@DuckChain_bot



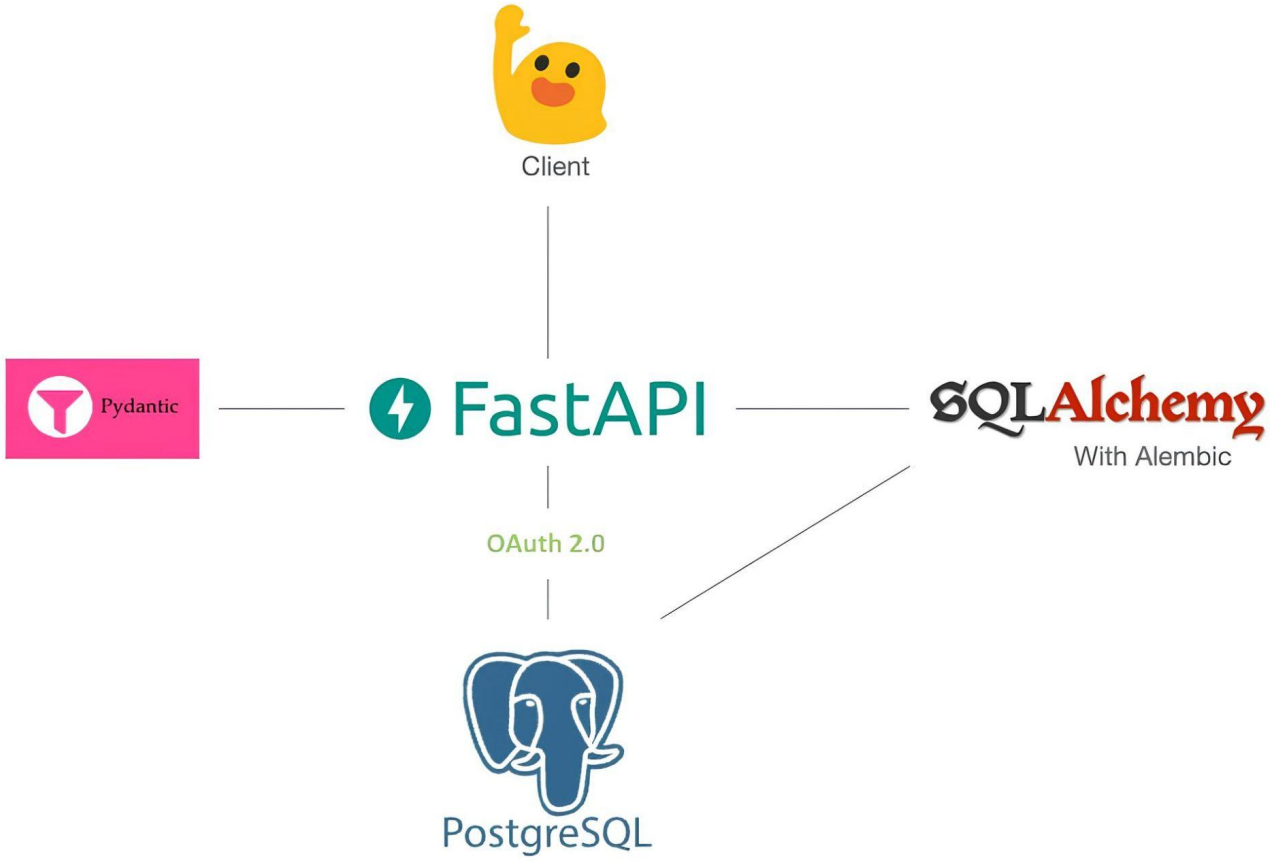
VS

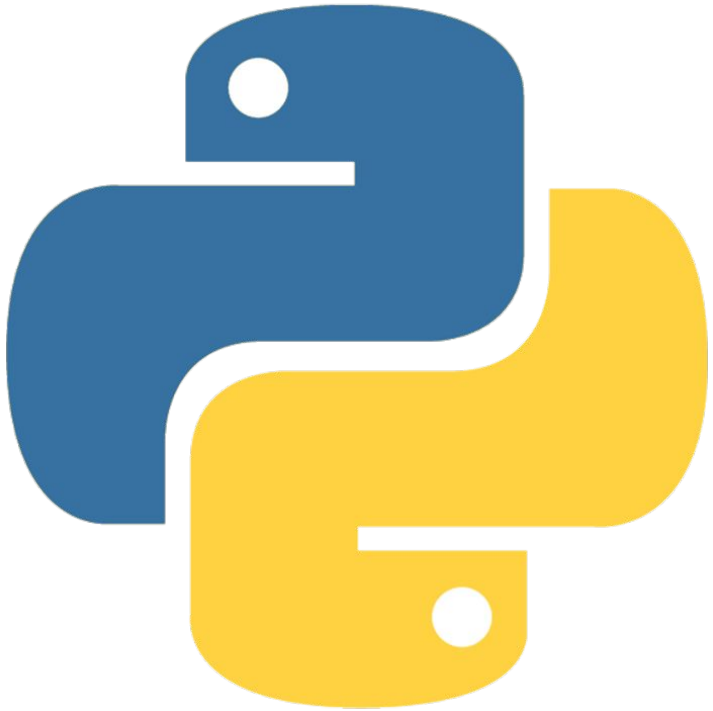
Функциональные требования

- Игровая механика кликера с накоплением ресурсов
- Система улучшений для повышения эффективности добычи ресурсов
- Коллекционирование NFT объектов, объединенных в тематические категории
- Система заданий с разными условиями выполнения и наградами
- Интеграция с кошельком TON для хранения цифровых активов
- Система ежедневных бонусов для стимулирования регулярного возвращения
- Социальные элементы: рейтинг, система рефералов

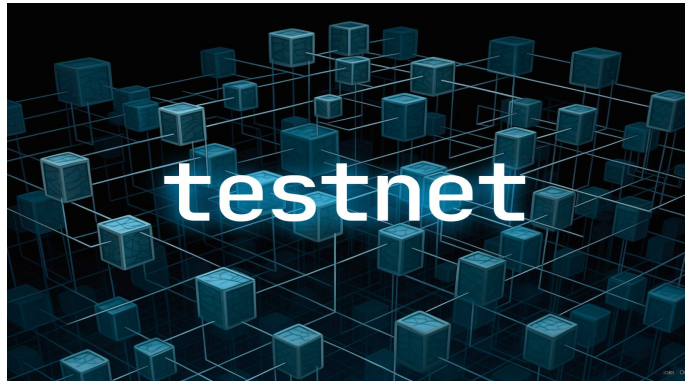
Выбор используемых в работе технологий

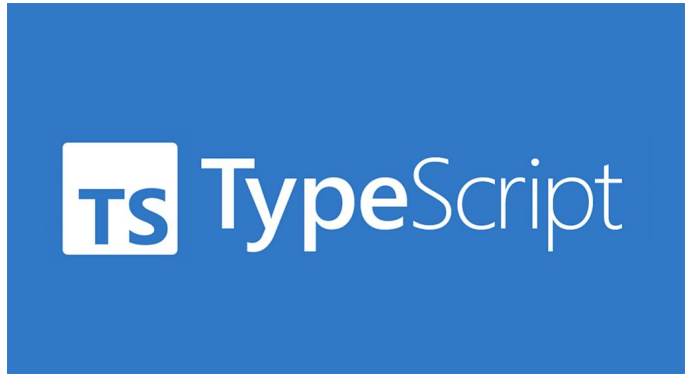
Выбор используемых в серверной части технологий





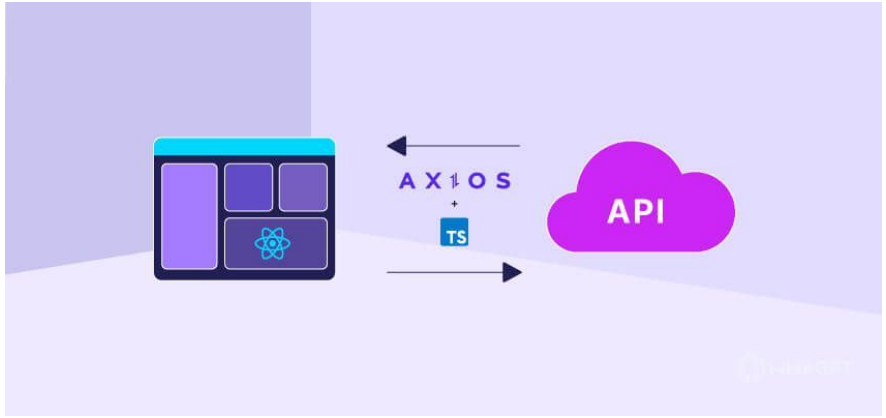
Выбор используемых технологий в смарт-контрактах





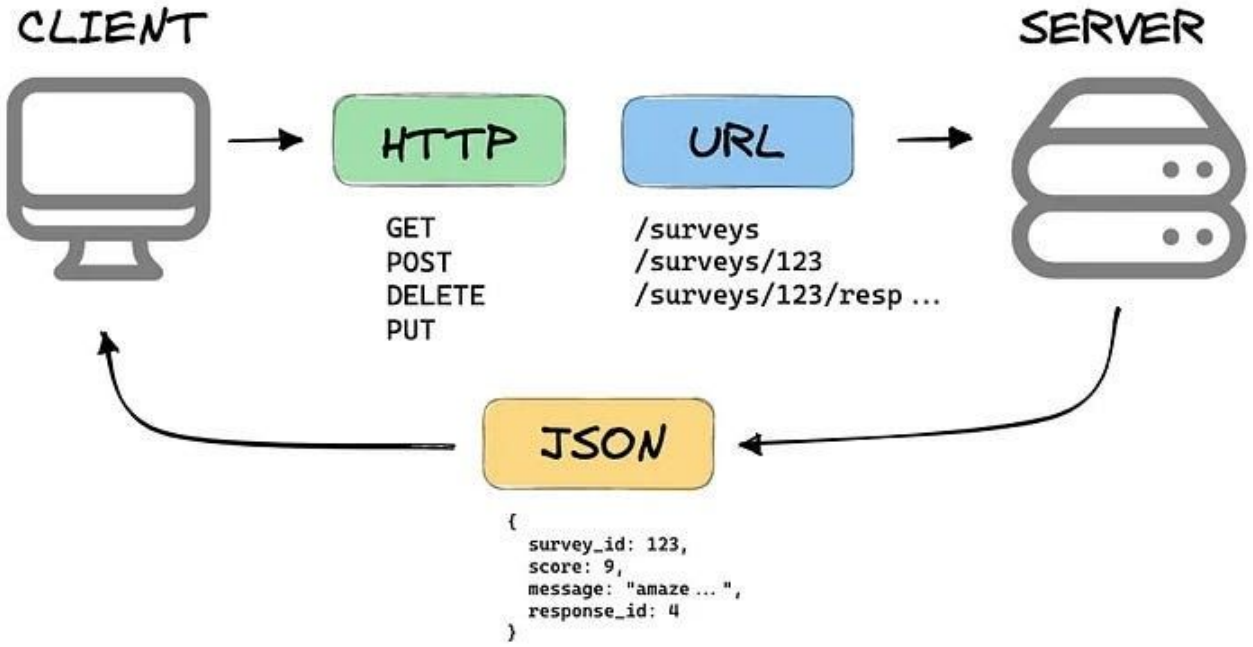
Выбор используемых в клиентской части технологий

React + Vite + TypeScript

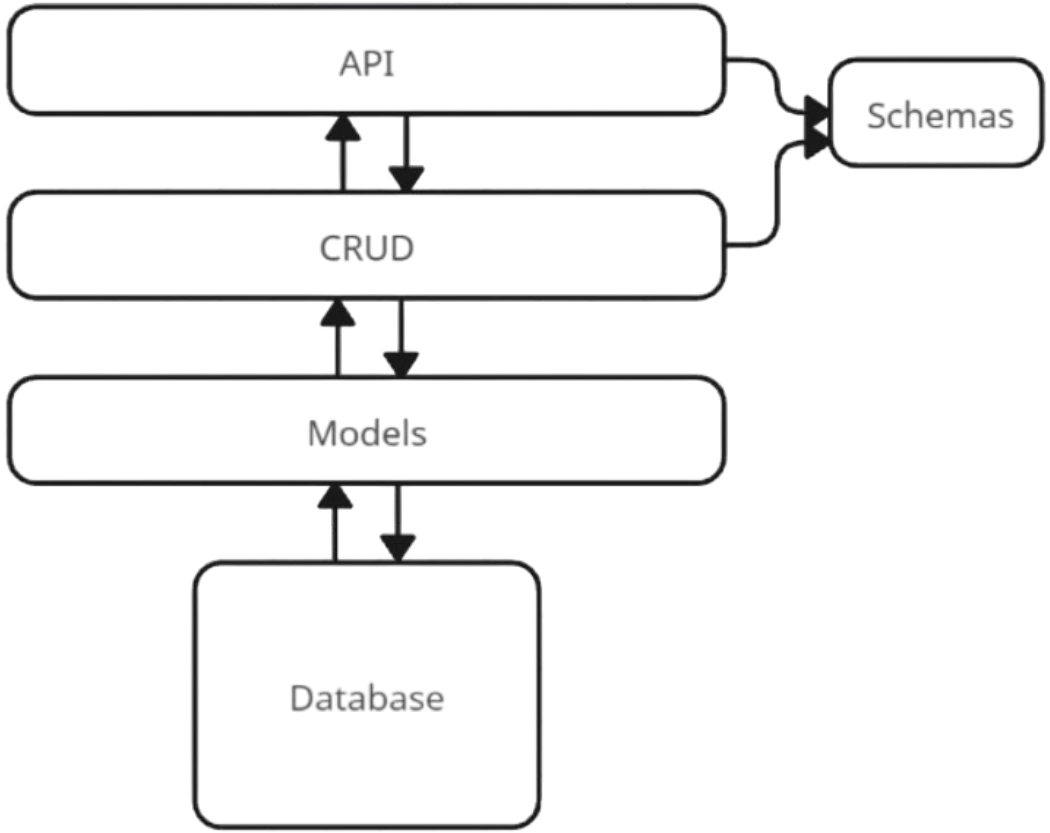


HTML CSS

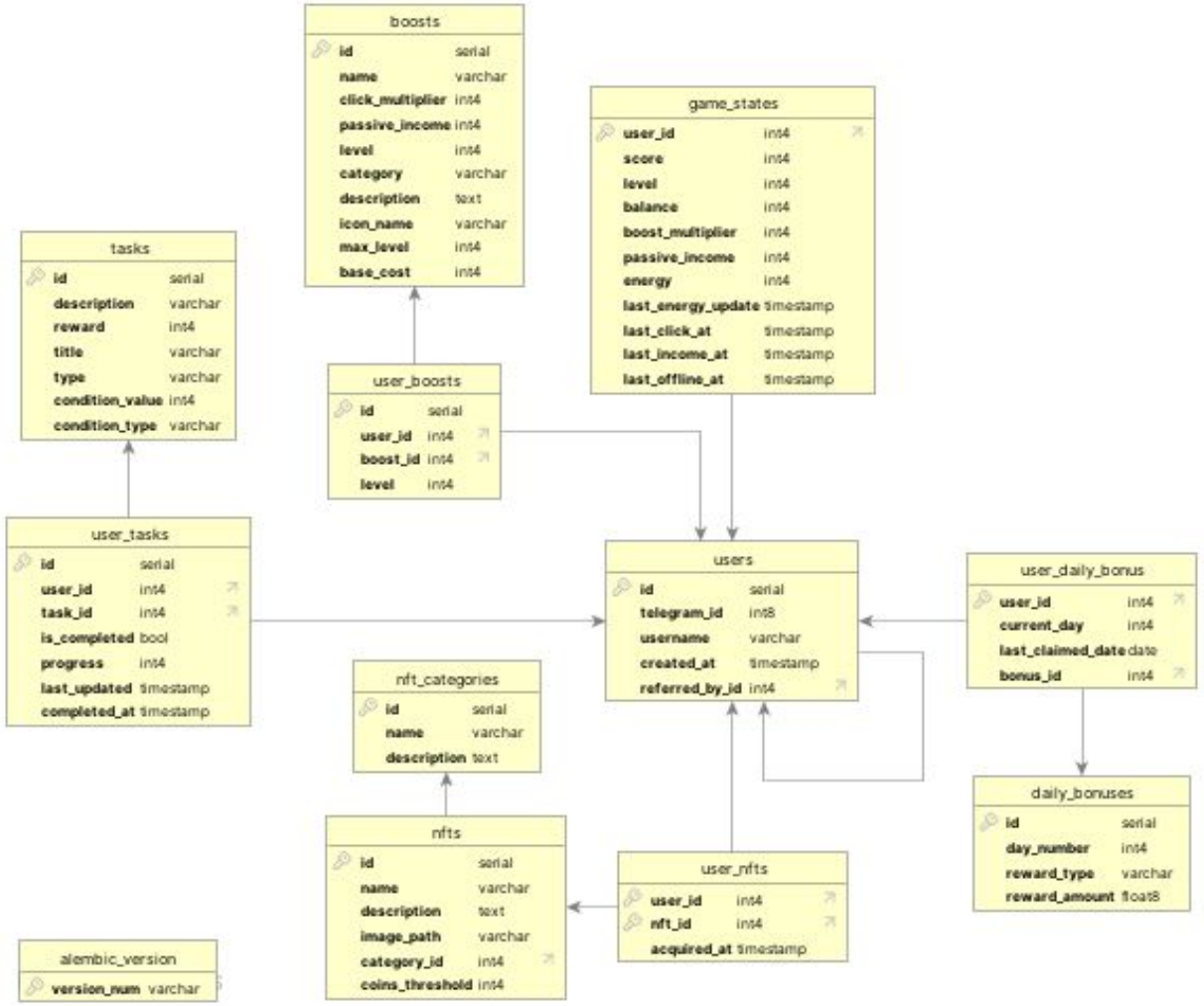
Архитектурный стиль



Структура серверной части



Структура базы данных



SpaceHunter API 1.0.0 OAS 3.1

/openapi.json

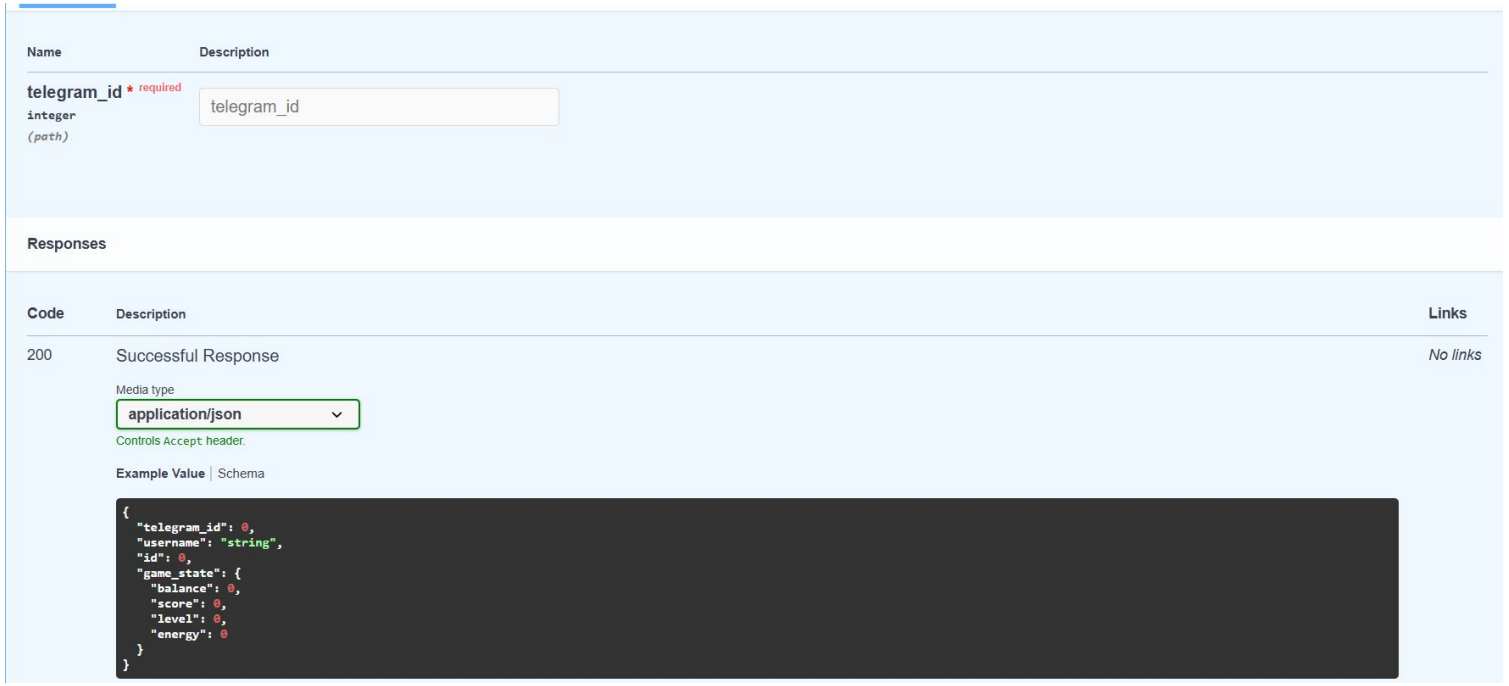
Бэкенд для Telegram Mini App с NFT и TON-кошельком

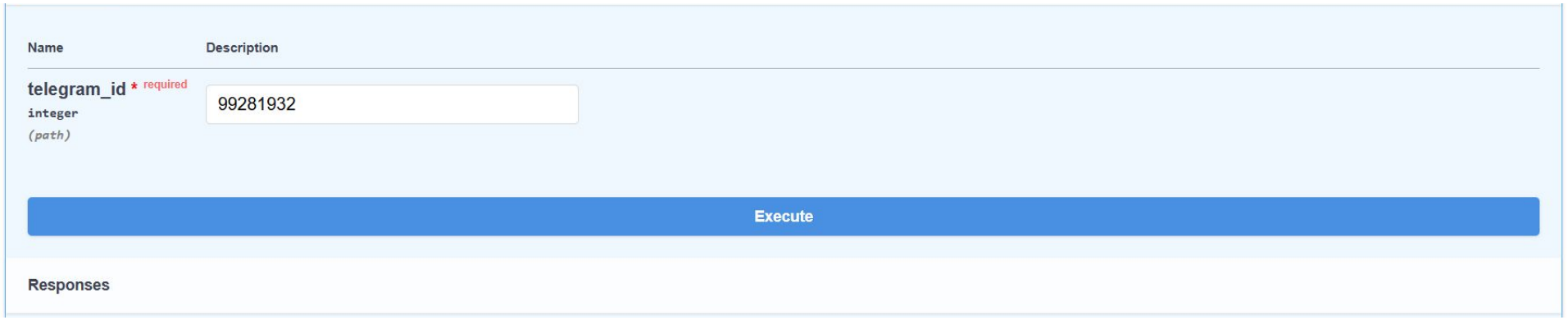
Users

- POST /api/users/register Register User
- GET /api/users/{telegram_id} Get User
- POST /api/users/register_with_referral Register User With Referral
- GET /api/users/referrals/{telegram_id} Get User Referrals

Boosts

- POST /api/boosts/upgrade Upgrade User Boost
- GET /api/boosts/ Read All Boosts
- GET /api/boosts/user/{telegram_id} Read User Boosts


Name
Description
telegram_id * required
integer
(path)
telegram_id
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"telegram_id": 0,
"username": "string",
"id": 0,
"game_state": {
"balance": 0,
"score": 0,
"level": 0,
"energy": 0
}
}


Name
Description
telegram_id * required
integer
(path)
99281932
Execute
Responses

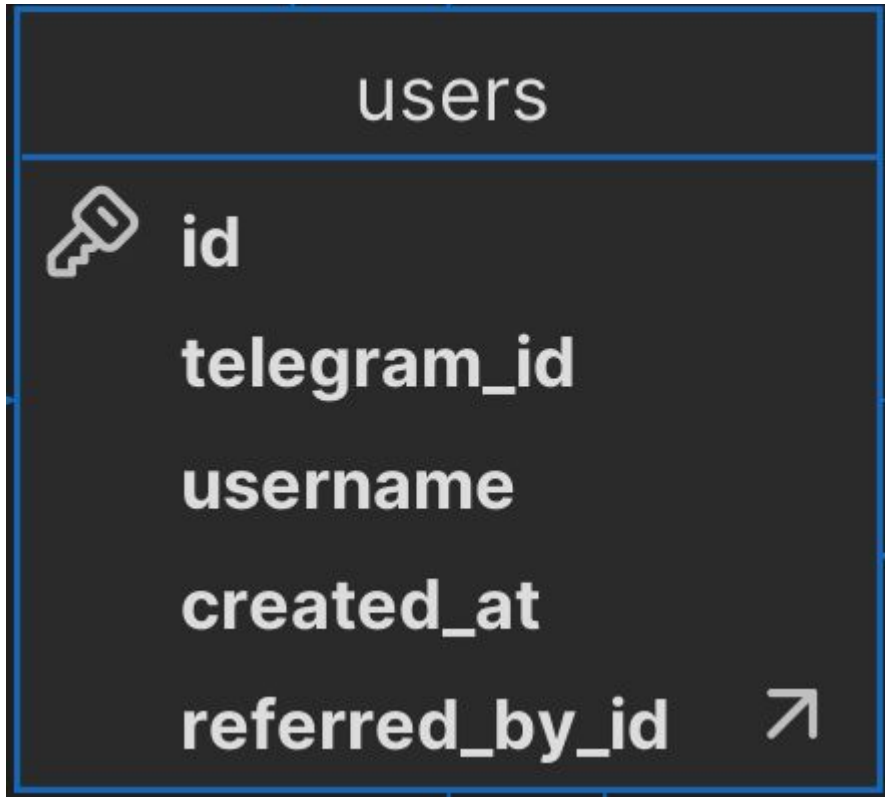
```python
# Схема для пользователя, который будет возвращен из базы данных
class UserWithGameState(UserBase):
    id: int
    game_state: GameStateBase

    class Config:
        orm_mode = True
```

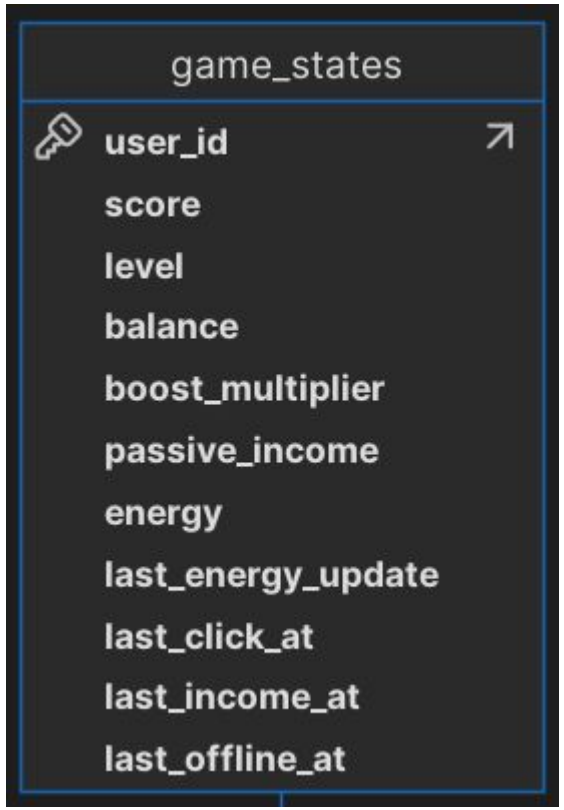
```python
class GameStateBase(BaseModel):
    balance: int
    score: int
    level: int
    energy: int

    class Config:
        orm_mode = True
```

```python
# Эндпоинт для получения пользователя по telegram_id
@router.get("/{telegram_id}", response_model=UserWithGameState)
async def get_user(telegram_id: int, db: AsyncSession = Depends(get_db)):
    user = await get_user_by_telegram_id(db, telegram_id)
    if user is None:
        raise HTTPException(status_code=404, detail="User not found")
    return user
```

```python
async def get_user_by_telegram_id(db: AsyncSession, telegram_id: int):
    result = await db.execute(
        select(User)
        .options(selectinload(User.game_state))
        .where(User.telegram_id == telegram_id)
    )
    return result.scalars().first()
```

Пользователь





telegram_id * required
integer
(path)

99281932

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/api/users/99281932' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/api/users/99281932
```

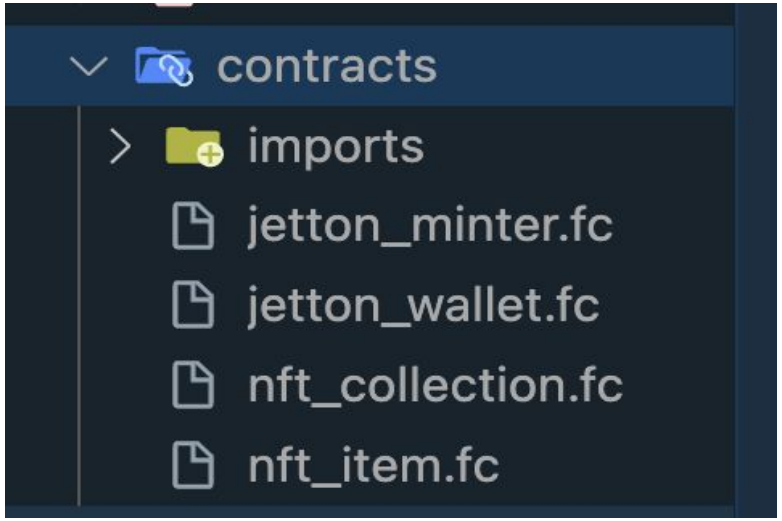
Server response

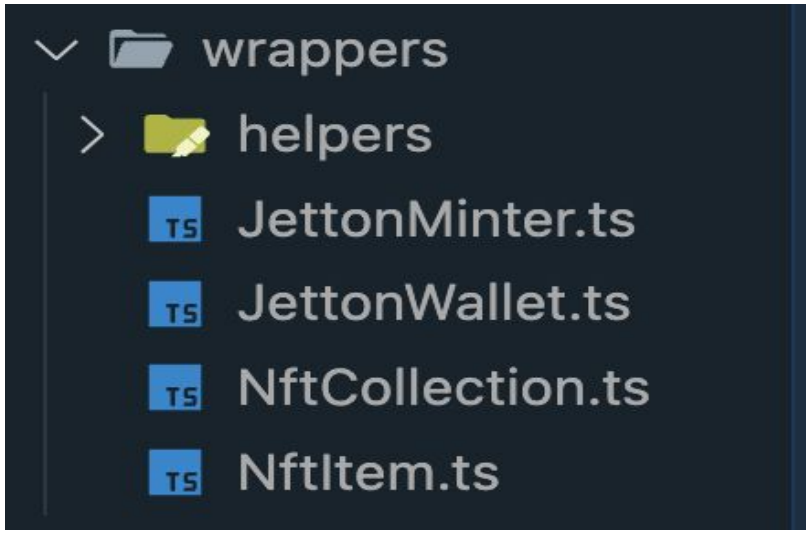
Code	Details
200	Response body

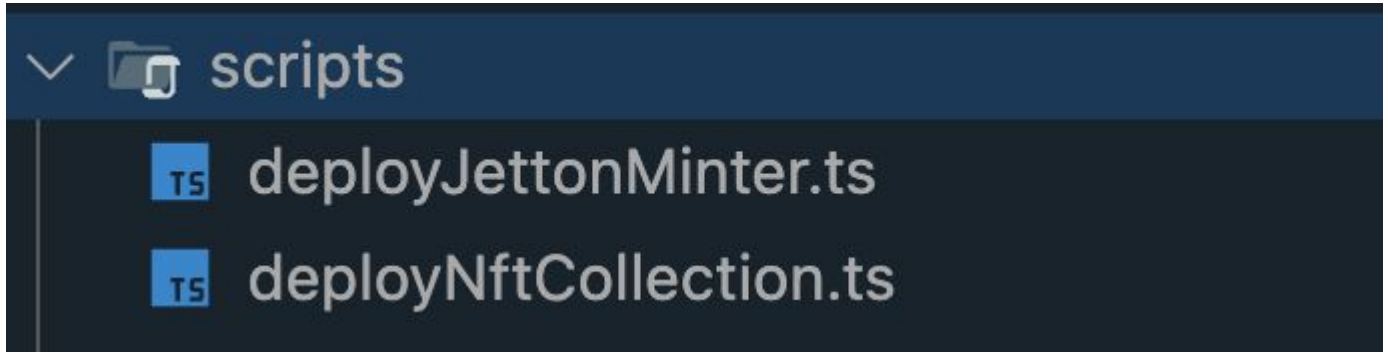
```
{
  "telegram_id": 99281932,
  "username": "dev_user_99281932",
  "id": 1,
  "game_state": {
    "balance": 440,
    "score": 402,
    "level": 6,
    "energy": 54
  }
}
```

Download

Структура смарт-контрактов







Основные методы смарт-контрактов

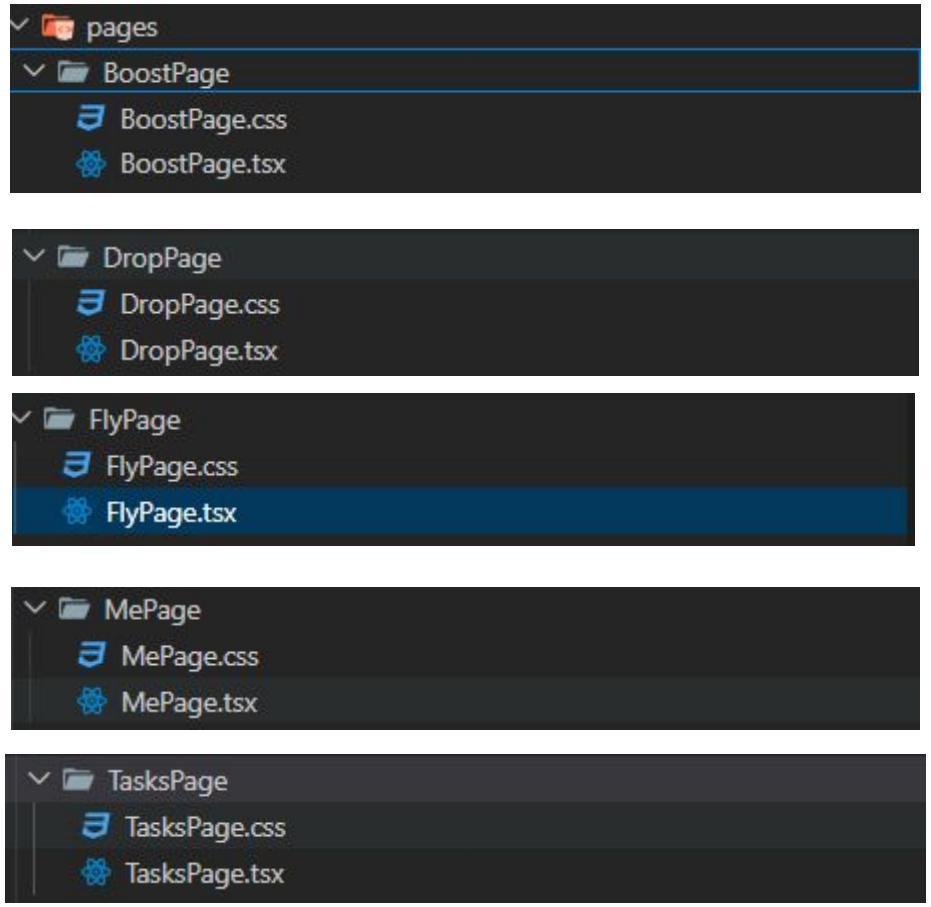
Внутренние сообщения recv_internal

```
() recv_internal(int my_balance, int msg_value, cell in_msg_full, slice in_msg_body) impure {
    if (in_msg_body.slice_empty?()) { ;; ignore empty messages
        return ();
    }

    slice cs = in_msg_full.begin_parse();
    int flags = cs~load_uint(4);
    if (flags & 1) {
        on_bounce(in_msg_body);
        return ();
    }

    slice sender_address = cs~load_msg_addr();
    cs~load_msg_addr(); ;; skip dst
    cs~load_coins(); ;; skip value
    cs~skip_bits(1); ;; skip extracurrency collection
    cs~load_coins(); ;; skip ihr_fee
    int fwd_fee = muldiv(cs~load_coins(), 3, 2);

    int op = in_msg_body~load_uint(32);

    if (op == op::transfer()) {
        send_tokens(in_msg_body, sender_address, msg_value, fwd_fee);
        return ();
    }

    if (op == op::internal_transfer()) {
        receive_tokens(in_msg_body, sender_address, my_balance, fwd_fee, msg_value);
        return ();
    }

    throw(0xffffff);
}
```

```
() recv_internal(int msg_value, cell in_msg_full, slice in_msg_body) impure {
    if (in_msg_body.slice_empty?()) { return (); } ;; ignore empty messages

    slice cs = in_msg_full.begin_parse();
    int flags = cs~load_uint(4);

    if (flags & 1) { ;; ignore bounced messages
        return ();
    }
    slice sender_adress = cs~load_msg_addr();

    int op = in_msg_body~load_uint(32);
    int query_id = in_msg_body~load_uint(64);

    (int total_supply, slice admin_adress, cell content, cell jetton_wallet_code) = load_data();

    if (op == op::mint()) {
        throw_unless(73, equal_slices_bits(sender_adress, admin_adress));
        slice to_adress = in_msg_body~load_msg_addr();
        int amount = in_msg_body~load_coins();
        cell master_msg = in_msg_body~load_ref();
        slice master_msg_cs = master_msg.begin_parse();
        master_msg_cs~skip_bits(32 + 64);
        int jetton_amount = master_msg_cs~load_coins();
        mint_tokens(to_adress, jetton_wallet_code, amount, master_msg);
        save_data(total_supply + jetton_amount, admin_adress, content, jetton_wallet_code);
        return ();
    }
}
```

Структура клиентской части

Одностраничное React приложение с компонентным подходом




```tsx
import {Route, Routes} from "react-router-dom";
import {FlyPage} from "@/pages/FlyPage/FlyPage.tsx";
import {BoostPage} from "@/pages/BoostPage/BoostPage.tsx";
import {TasksPage} from "@/pages/TasksPage/TasksPage.tsx";
import {MePage} from "@/pages/MePage/MePage.tsx";
import {DropPage} from "@/pages/DropPage/DropPage.tsx";
import {ErrorPage} from "@/pages/ErrorPage/ErrorPage.tsx";


export function AppRouter() {
    return (
        <Routes>
            <Route path="/SpaceHunter/fly" element={<FlyPage />} />
            <Route path="/SpaceHunter/boost" element={<BoostPage />} />
            <Route path="/SpaceHunter/tasks" element={<TasksPage />} />
            <Route path="/SpaceHunter/me" element={<MePage />} />
            <Route path="/SpaceHunter/drop" element={<DropPage />} />
            <Route path="/SpaceHunter/error" element={<ErrorPage />} />
            <Route path="*" element={<FlyPage />} />
        </Routes>
    )
}
```

Структура клиентской части

Объединение компонентов:

App.tsx

```tsx
export function App() {
  const lp = useLaunchParams();
  const isDark = useSignal(miniApp.isDark);

  return (
    <BrowserRouter>
      <AppRoot
        appearance={isDark ? 'dark' : 'light'}
        platform={['macos', 'ios'].includes(lp.platform) ? 'ios' : 'base'}
      >
        <AuthWrapper>
          <PageHeader />
          <AppRouter />
          <NavigationBar />
        </AuthWrapper>
      </AppRoot>
    </BrowserRouter>
  );
}
```

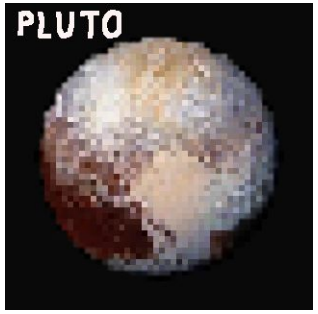
```tsx
export function Root() {
  return (
    <ErrorBoundary fallback={ErrorBoundaryError}>
      <TonConnectUIProvider
        manifestUrl={publicUrl('tonconnect-manifest.json')}
      >
        <App/>
      </TonConnectUIProvider>
    </ErrorBoundary>
  );
}
```

```tsx
const root = ReactDOM.createRoot(document.getElementById('root')!);

try {
  // Configure all application dependencies.
  init(retrieveLaunchParams().startParam === 'debug' || import.meta.env.DEV);

  root.render(
    <StrictMode>
      <Root/>
    </StrictMode>,
  );
} catch (e) {
  root.render(<EnvUnsupported/>);
}
```

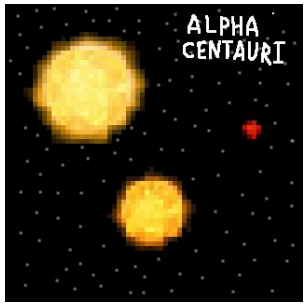
Клиентская часть: ассеты

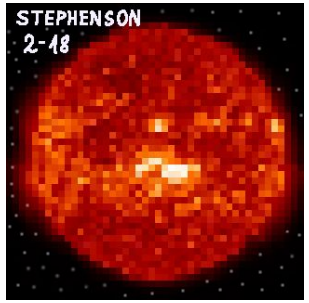
NFT, получаемые за игровой прогресс: 112 штук

Солнечная система

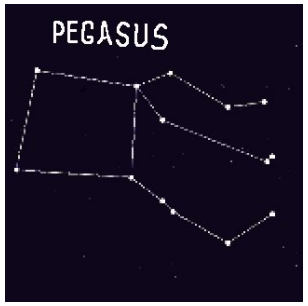


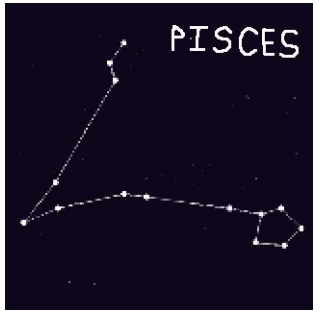
Звёзды





Созвездия

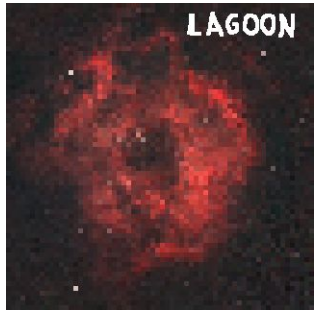


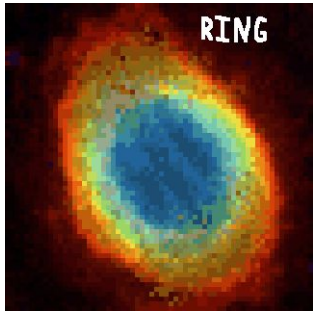


Клиентская часть: ассеты

NFT, получаемые за игровой прогресс: 112 штук

Туманности





Чёрные дыры

Галактики

Клиентская часть: ассеты

NFT, получаемые за игровой прогресс: 112 штук

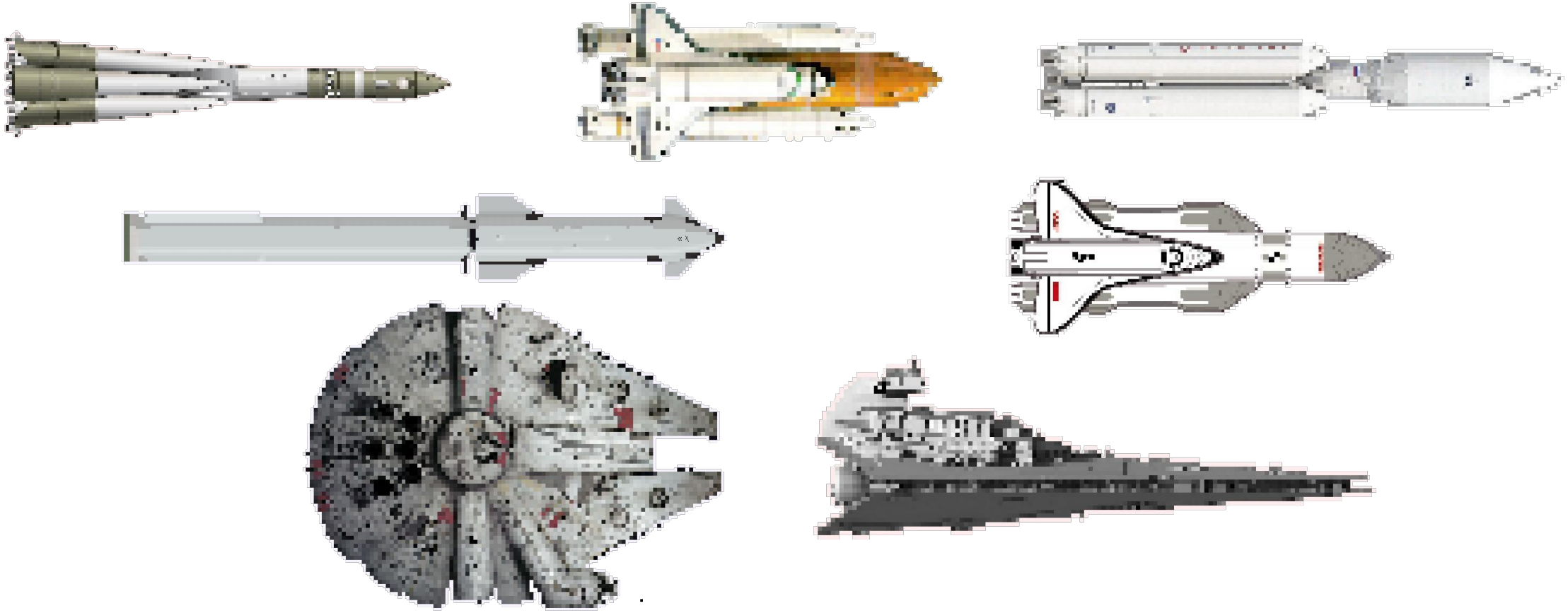
Кино Отсылки

Музыкальные Отсылки

Бонусные

Клиентская часть: ассеты

Космические корабли, распределенные по уровням

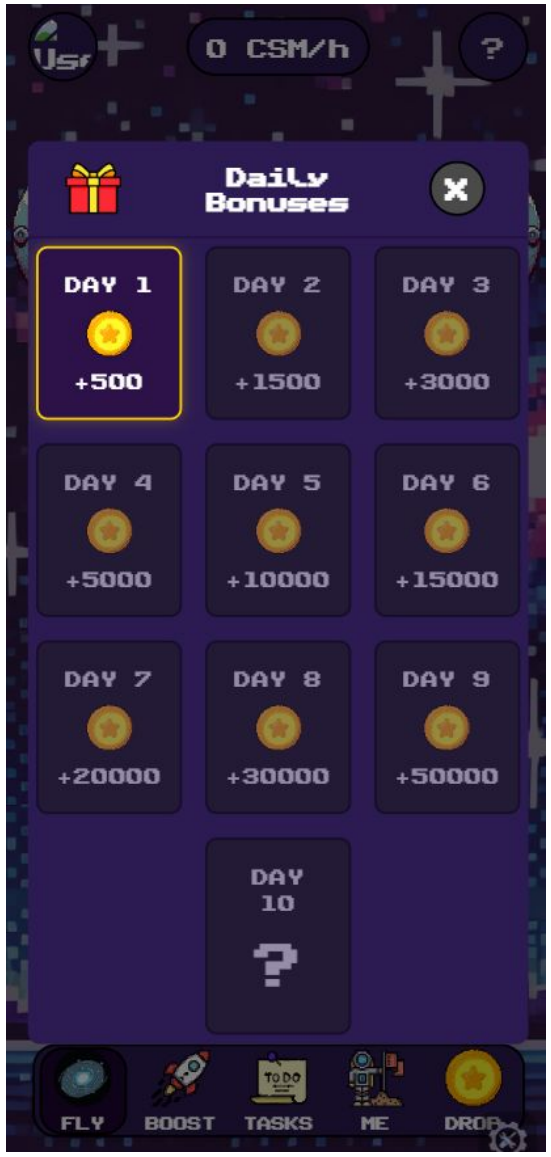


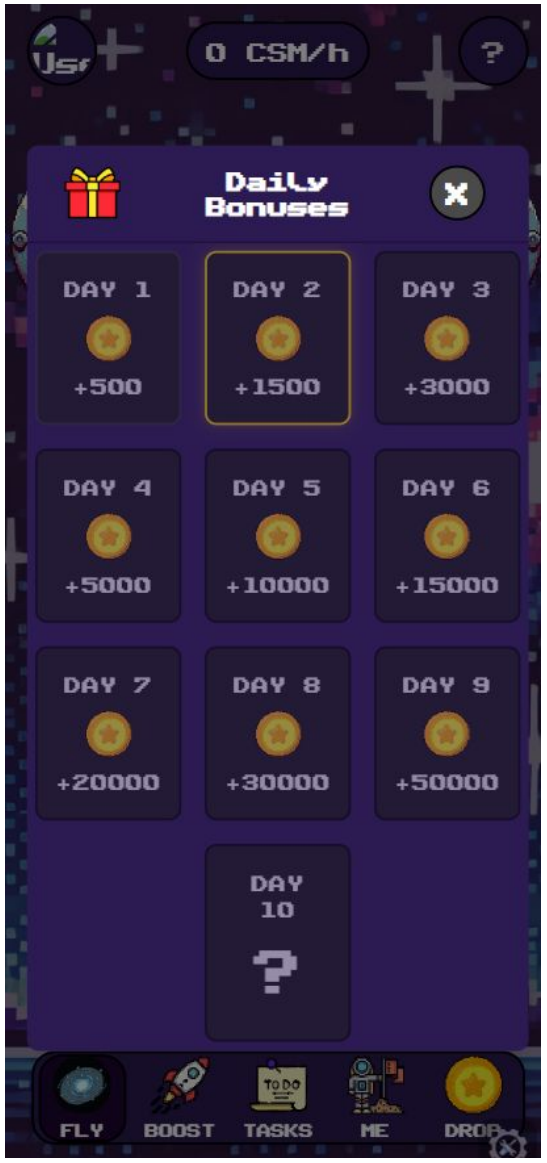
Клиентская часть: страница Fly

Раздел приложения с основными игровыми механикой

Клиентская часть: страница Fly

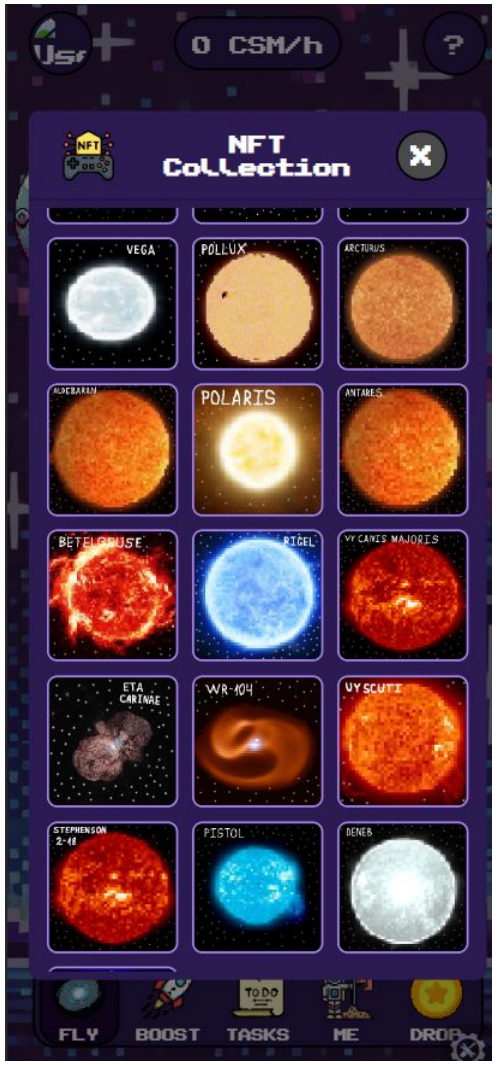
Модальное окно ежедневных бонусов

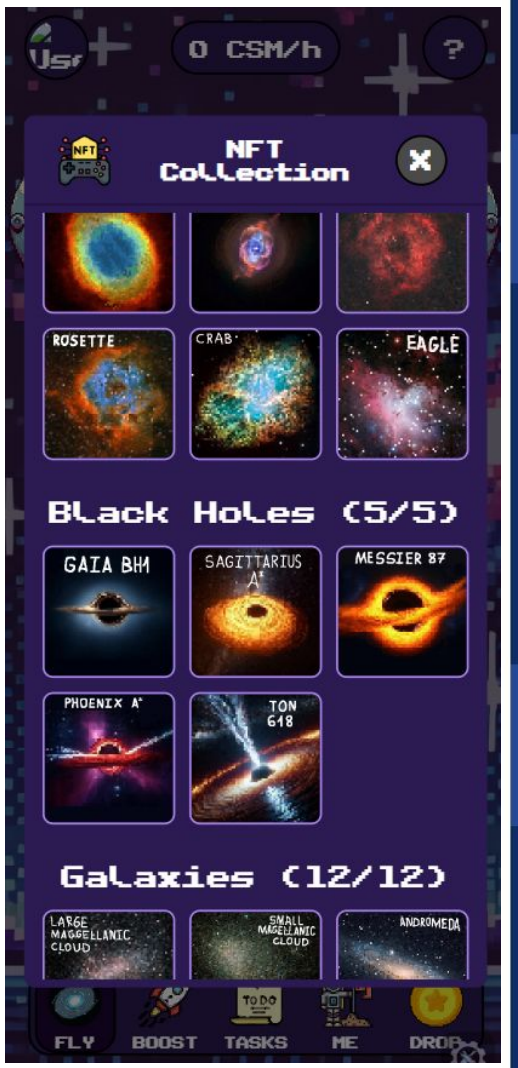




Клиентская часть: страница Fly

Модальное окно просмотра полученной NFT коллекции



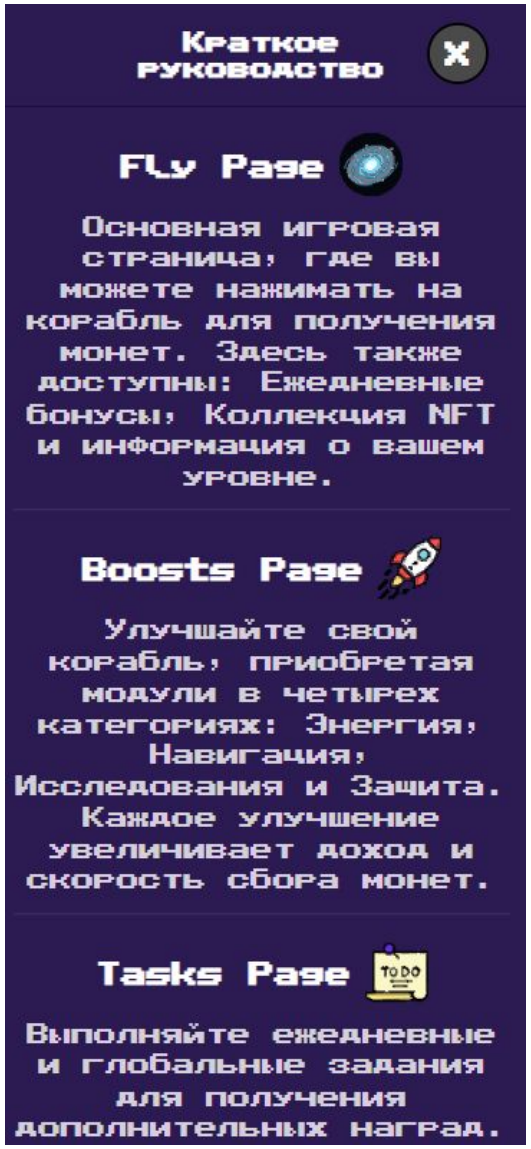


Клиентская часть: страница Fly

Модальное окно просмотра текущего уровня игры

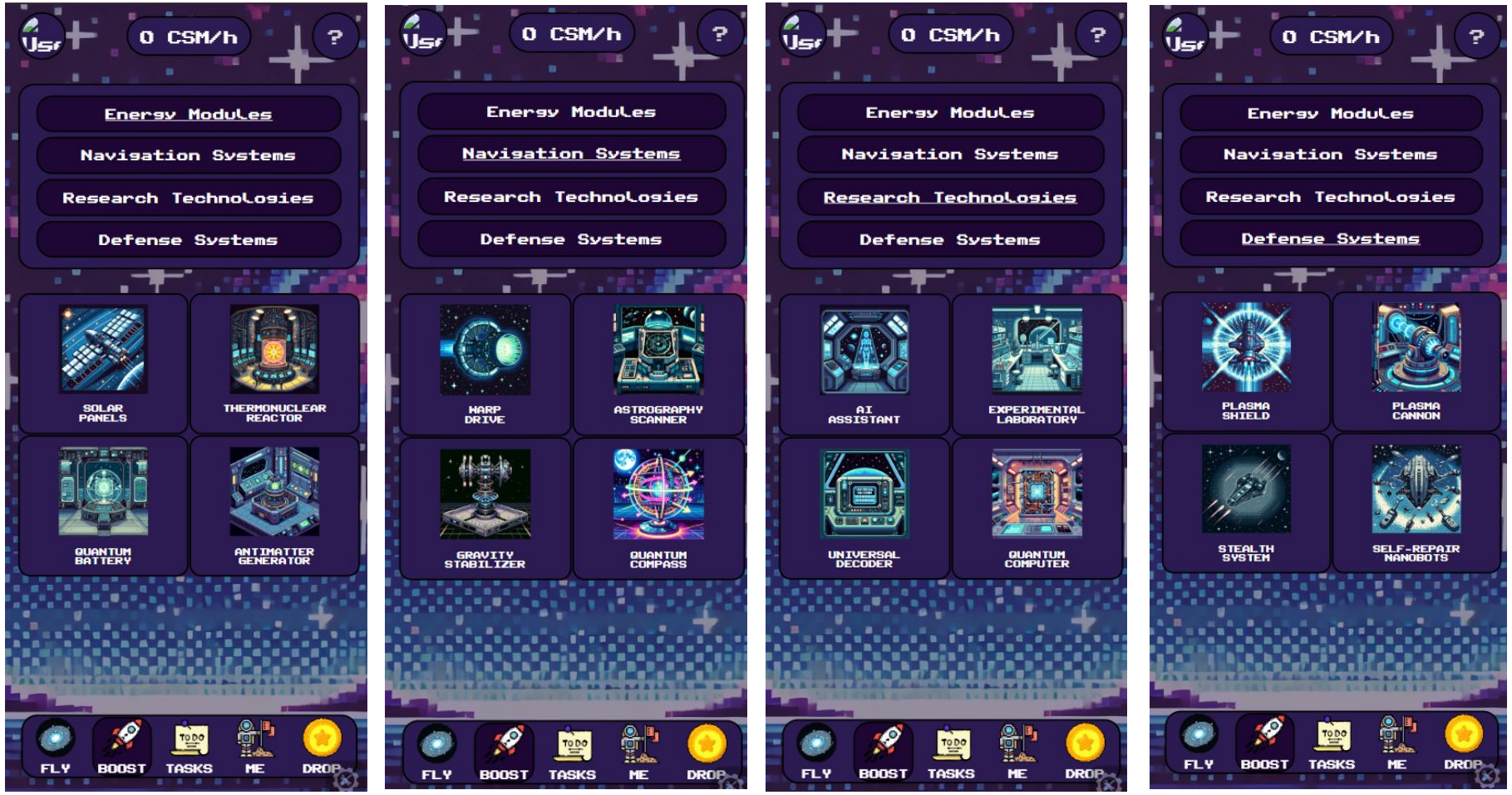
Клиентская часть: страница Fly

Модальное окно краткого руководства по игре



Клиентская часть: страница Boosts

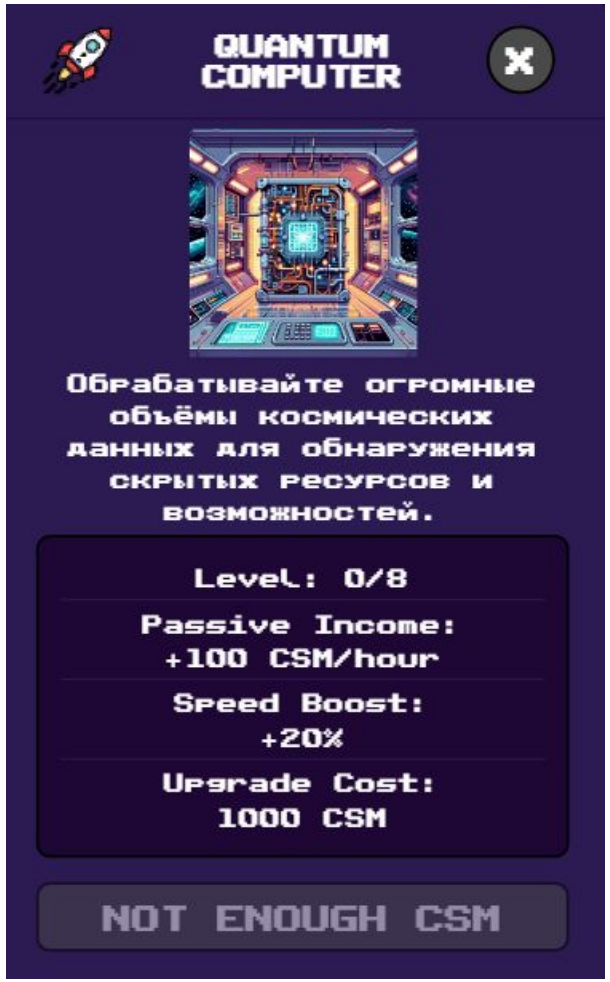
Раздел приложения с покупкой улучшений



Клиентская часть: страница Boosts

Модальное окно конкретного улучшения

Показываются основные бонусы от покупки и цена



Клиентская часть: страница Tasks

Модальное окно выполнения заданий

Два вида: восстанавливающиеся ежедневные и глобальные одноразовые

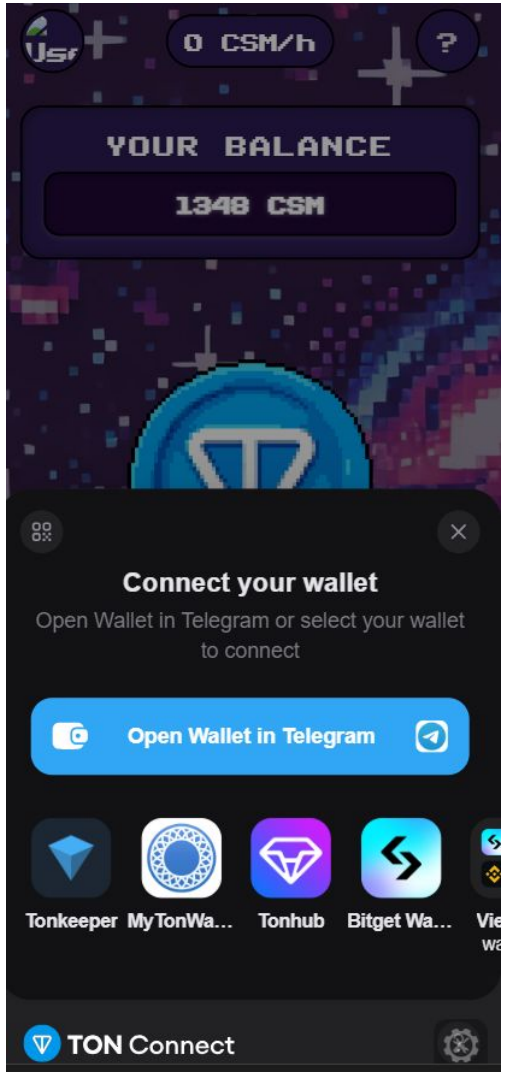
Клиентская часть: страница Me

Раздел приложения по приглашению новых пользователей

Два способа отправки реферальной ссылки. Возможность отслеживать прогресс друзей

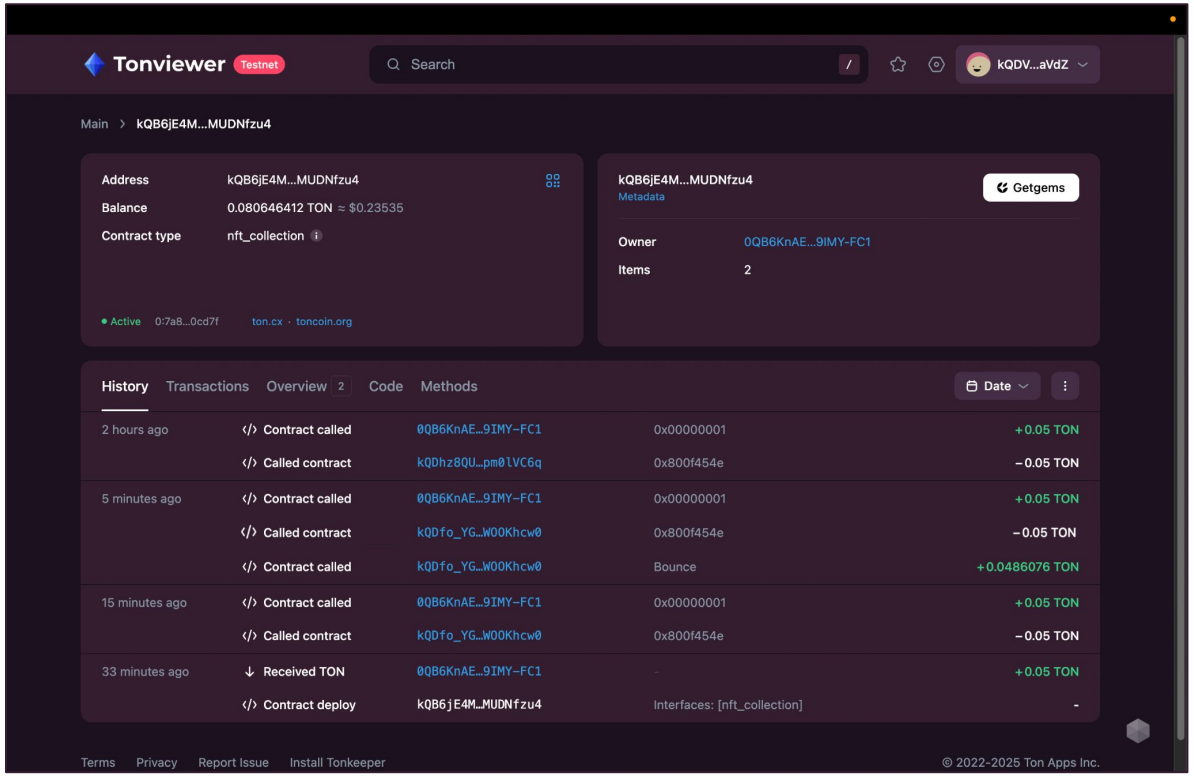
Клиентская часть: страница Drop

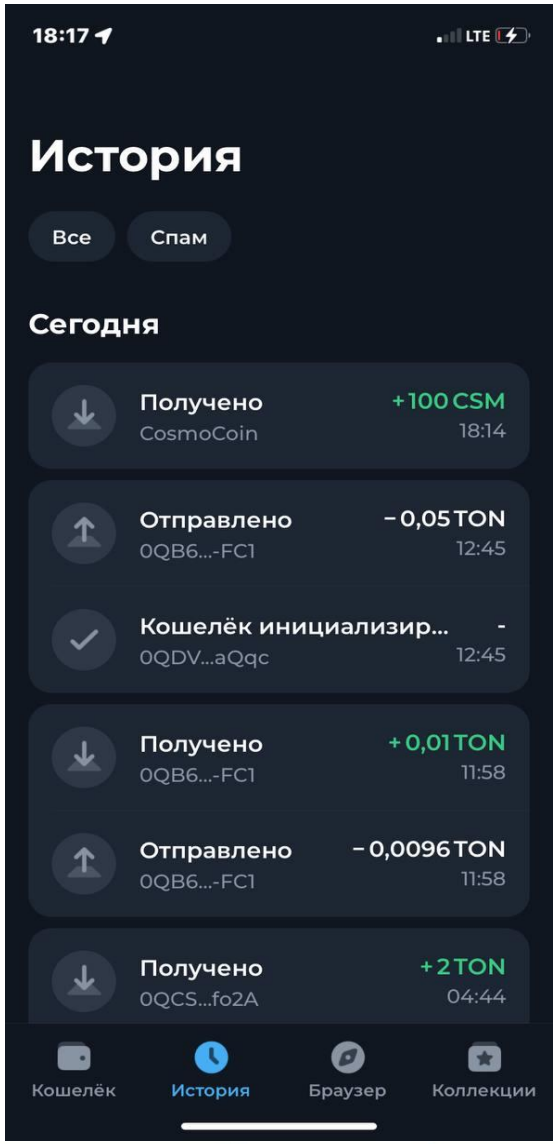
Раздел приложения по подключению TON кошелька

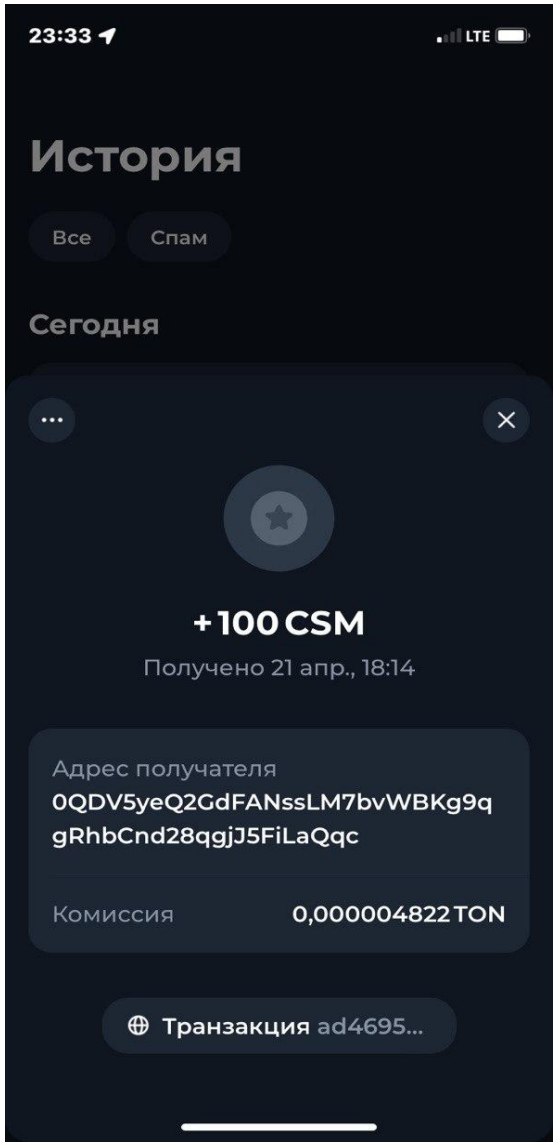


Смарт-контракты: Airdrop

При airdrop пользователям отправляются токены CosmoCoin и NFT







Результаты работы

1. Проведен анализ аналогов
2. Разработано полноценное веб-приложение, интегрированное в Telegram Mini Apps с кликерной механикой и NFT коллекционированием
3. Реализованы все три части проекта: фронтенд, бэкенд и смарт-контракты
4. Создана система прокачки и заданий
5. Приложение демонстрирует объединение привычных игровых механик с Web3

Направления дальнейшей работы

- Расширение коллекции NFT новыми категориями и объектами
- Внедрение социальных механик: торговля между игроками, совместные задания, глобальный рейтинг
- Интеграция с внешними маркетплейсами для торговли полученными NFT
- Геймификация образовательного контента о космосе
- Расширение функциональности TON-кошелька для более глубокой интеграции с экосистемой блокчейна (интеграция с MainNET)
- Дополнительные возможности встроенных покупок (Telegram Stars)

Демонстрация

Спасибо за внимание!


ВЫСШАЯ • ШКОЛА • ЭКОНОМИКИ •