Introduction

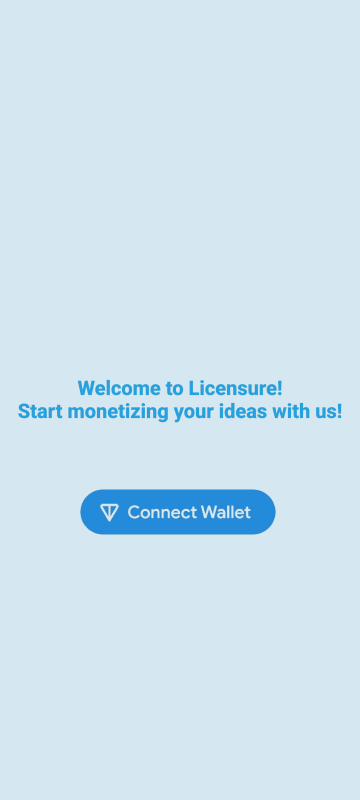
This document describes the mapping of the user interface to backend methods.

Pages

1. **Authorization page.**

**Description**: The first page when the web application loads. There is an option for authorization in the application only through one of the TON wallets.

**Mockup**:



**Implementation**: Button “Connect Wallet” works through the “**@tonconnect/ui-react**” library. To add this button, you must use the **<TonConnectButton>** component. In addition, the React application must be “wrapped” in the **<TonConnectUIProvider>** component.

After authorization via TON wallet, need to automatically create a user account in the system.

To create a user in the system must use the following backend endpoint “**Create User**”:

**POST /api/createUser  
Host:** <https://licensure.tech> **Headers:  
 1. Content-Type: application/json  
 2. Authorization: Bearer {Token Value}  
Body:**{

"wallet\_address": {TON Wallet Address}

}

**Response Example**:

**OK** – 201 CREATED:

{

"nickname": "HappyKitten5407",

"user\_id": 6,

"wallet\_address": {TON Wallet Address}

}

**User already exist** – 500 INTERNAL SERVER ERROR:

{

"error": "User with this wallet address already exists"

}

**Empty Body** – 400 BAD REQUEST.

**Empty Parameter** – 400 BAD REQUEST:

{

"error": "Wallet address are required"

}

Each registered user has two unique parameters in addition to the wallet address:

**user\_id** – The user's sequence number in the system, auto-incrementing.

**nickname** – Unique nickname of the user in the system, which can be used to display instead of the wallet address.

1. **P2P Market page.**

**Description**: After authorization and user creation, the system should display a P2P marketplace page where the user can view all offers and purchase a license.

**Mockup**:



**Implementation**: To retrieve data about all licenses, you must use the smart contract method, which accesses the TON blockchain network and returns all licenses that the smart contract stores:

import { useAsyncInitialize } from './useAsyncInitialize';

import { useTonClient } from './useTonClient';

import { Main } from '../../wrappers/Main';

import { Address, OpenedContract } from 'ton-core';

const {client} = useTonClient();

const mainContract = useAsyncInitialize(async() => {

if (!client) return;

const contract = Main.fromAddress(Address.parse('{Address of contract in blockchain}'));

return client.open(contract) as OpenedContract<Main>;

}, [client]);

Now you can refer to different contract **GET** methods:

mainContract?.getArrayOfLicenses()

**Response Example**:

{

'$$type': 'LicenseArray',

map: Dictionary {

\_key: {

bits: 257,

serialize: [Function: serialize],

parse: [Function: parse]

},

\_value: { serialize: [Function: serialize], parse: [Function: parse] },

\_map: Map(1) {

'b:52386475024439176110681937689299597561070022963244119515134366198178265804108' => [Object]

}

},

length: 1n

}

This object contains three main child objects:

**'$$type'** – The type of message you sent to the blockchain network.

**map** – A dictionary that contains all the licenses that are stored in the smart contract.

**length** – Total number of licenses in the dictionary.

In order to get all the necessary license parameters, it is needed to refer to the \_map.Map(n) {} attribute in map object:

Map(1) {

'b:9302427541248522615927265473332735202783708273626430849222715083863515990000' => {

'$$type': 'License',

licenseId: 9302427541248522615927265473332735202783708273626430849222715083863515990000n,

sellerAddress: EQBGhqLAZseEqRXz4ByFPTGV7SVMlI4hrbs-Sps\_Xzx01x8G,

buyerAddress: EQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAM9c,

createdAt: 1715385981n,

contentName: 'Videos with dogs',

contentDescription: 'super!!!',

contentUrls: '1 - https://docs.tact-lang.org/cookbook/data-structures',

licenseType: 'Restricted license',

contentCategory: 'Video',

contentSubcategory: 'Internet video',

price: 50n,

currency: 'TON',

allRestrictions: 'Duration: 1 year; Purpose: Training neural networks, Marketing; Modification: No',

additionalTerms: '',

status: 'Pending'

}

}