Justicket

CAHIER DES CHARGES

La société **Justicket** est une société nouvellement créée et qui souhaite se positionner comme une plateforme de gestion dématérialisée de billetterie.

Le métier de la billetterie a deux grands challenges actuellement :

1. *La lutte contre les faux billets*
2. *Le fait que le marché secondaire - la revente de particulier à particulier - soit en même temps non fiable pour les acheteurs et que les salles de concerts n'en bénéficient pas.*

Justicket souhaite proposer une solution aux salles de concerts se basant sur l'utilisation de la technologie tokenization sur blockchain pour répondre à ces enjeux.

Elle souhaite lancer son MVP - Minimum Viable Product - en Juillet 2023, en partenariat avec le Zenith de Lille.

Une première étape importante est la réalisation d'un POC - Proof Of Concept - fonctionnel pour **le 30 Mars 2023.** Cette première étape vise à démontrer **la logique d'émission et gestion des billets**. Les billets correspondent uniquement à du placement libre.

Le MVP ajoutera la dimension de vérification des billets à l'entrée des salles, ainsi que la gestion du marché secondaire. La vente de places numérotées sera également possible.

Le CTO a présélectionné des prestataires pour réaliser les différents projet, ainsi que les technologies qui pourraient servir à constituer le SI. Dans la mesure du possible, la DSI préfèrera les solutions gratuite et soumises à licence open source, ou les solutions SaaS freemium.

Les livrables attendus sont :

- Pour chaque projet, un repository Github public, contenant **le code source, les tests et toute documentation utile au format Markdown**
- Pour l'ensemble des projets, une URL permettant d'accéder à une version du POC déployée, qui servira à la recette du client

Voici les projets identifiés :

1) **Application API de gestion des concerts**
2) **Application de tokenization des billets**
3) **Application Web utilisateur**
4) **Application dashboard**

L'application dispose de 3 rôles fonctionnels:

- **Administrateur** (administrator) **:** Utilisateur ayant le droit de créer des entités dans la base de données de l'API. Dans le cadre du POC, il s'agit d'une liste d'adresses statiques, connues préalablement.
- **Acheteur** (buyer) **:** Utilisateur connecté avec son wallet à l'application, ayant déjà ou non effectué un premier achat.
- **Anonyme** (anonym) : Utilisateur non connecté

Le terme « **utilisateur** » désigne, indistinctement, n'importe lequel de ces rôles.

Le terme « **système** » désigne, le système informatique, indistinctement du service concerné.

Dans la description des projets le terme « **prestataire** » désigne la société en charge de la réalisation du projet, le terme « **client** » désigne la société Justicket.

1) Projet 1 : Application API de gestion des concerts

a. Description

Cette application a pour objectif de gérer la base de donnée des concerts. Il s'agit d'une application web.

La réalisation de ce lot est confié par le client à un prestataire.

b. Stories

Story 1 : En tant qu'**Administrateur**, je peux **créer un nouveau concert**, afin de **créer une entrée dans le système et de minter NFT représentant les billets**

Story 2 : En tant qu'**Administrateur**, je peux **supprimer concert existant**, afin de **supprimer une entrée dans le système et de burn NFT représentant les billets non vendus**

Story 3 : En tant qu'**Administrateur**, je peux **modifier concert existant**, afin de **modifier une entrée dans le système, tenant compte du fait que le nombre de places n'est pas modifiable.**

Story 4 : En tant qu'**Utilisateur**, je peux **lister l'ensemble des concerts disponibles**, afin de **connaitre le catalogue de concerts.**

Story 5 : En tant qu'**Utilisateur**, je peux **lire l'ensemble des données d'un concert,** afin d'en **connaitre le détail.**

c. Règles de gestion

RG1 : une requête d'API provient d'un administrateur, si et seulement si elle a été signée avec une adresse d'un administrateur.

RG2 : les suppressions de données sont toujours des suppressions logiques

RG3 : les identifiants techniques en base de données doivent être au format UUID v4

d. Limites pour le POC du logiciel

L'ajout d'administrateurs s'effectue directement par requête SQL par l'administrateur de la base de données.

e. Technologies retenues

i. Démarche

L'application est une application backend qui expose une API au format REST/JSON.
Une spécification OPEN API est à fournir le plus rapidement possible aux projets en ayant besoin.

ii. Framework et langage

L'application doit être réalisée en Typescript et exécutée à l'aide du runtime Node.js, en version 18 ou supérieure.

Le prestataire est libre d'utiliser ou non un framework, bien que cela soit encouragé par le client.

Les frameworks acceptés par le client sont :

- Express, si vous choisissez d'utiliser un framework « unopinionated » https://expressjs.com/
- NestJS, si vous choisissez d'utiliser un framework « opinionated » https://docs.nestjs.com/

iii. Base de données

La base de données à utiliser est PostgreSQL.

Le prestataire est libre d'utiliser ou non un ORM, bien que cela soit encouragé par le client.

La solution typeORM https://typeorm.io/ en mode DataMapper est à privilégier le cas échéant.

iv. Utilitaire

La librairie taquito https://tezostaquito.io/ peut être utilisée pour la vérification des signature ou les interactions avec la blockchain Tezos.

v. Metadonnées

La création d'un nouveau token nécessite la création de métadonnées. Les métadonnées doivent être stockées sur IPFS. On utilisera le service Pinata Cloud https://docs.pinata.cloud pour attacher (**pin**) le fichier JSON au réseau IPFS. Le niveau gratuit permet la réalisation du POC.

vi. URIs

URL du noeud RPC Tezos : https://ghostnet.tezos.marigold.dev *nécessaire pour configurer le TezosTookit de taquito*

Documentation d'API Tezos : https://api.marigold.dev/general/ghostnet *à priori vous ne devriez pas avoir à vous en servir directement si vous utiliser Taquito*

2) Projet 2 : Application de tokenization des billets

a. Description

Cette application a pour objectif de gérer les NFT de billets. Il s'agit de smart contracts.

La réalisation de ce lot est confié par le client à un prestataire.

b. Stories

Story 1 : En tant qu'**Administrateur**, je peux **minter une collection de NFT**, afin de **représenter les billets de mon concert.**

Story 2 : En tant qu'**Acheteur**, je peux **acheter un billet**, afin d'**assister au concert**

Story 3 : En tant qu'**Utilisateur**, je peux **transférer un billet que je possède**, afin **de donner mon billet à un autre utilisateur**

Story 4 : En tant qu'**Utilisateur**, je peux **burn un billet que je possède**, afin **de signifier mon entrée dans le concert.**

Story 5 : En tant qu'**Administrateur**, je peux **retirer (withdraw) l'argent du compte détenu sur le contrat de mon concert**, afin **d'être payé sur mon adresse**

Story 6 : En tant qu'**Administrateur**, je peux **je peux annuler un concert**, afin d'**annuler l'évènement et de rembourser les acheteurs**

c. Règles de gestion

RG1 : Lorsqu'un billet est **minté**, il appartient à l'administrateur du concert

RG2 : Acheter un billet revient à payer EXACTEMENT le prix du billet en **tez** ou **mutez** et à transférer le billet à l'acheteur. Le paiement est stocké dans la balance du smart contract du concert.

RG3 : Burn un billet revient à transférer le token à l'adresse **tz1burnburnburnburnburnburnburjAYjjX** https://tzkt.io/tz1burnburnburnburnburnburnburjAYjjX

RG4 : les billets sont des tokens sur la blockchain Tezos qui respectent le standard FA2 définit dans le TZIP 12 https://tzip.tezosagora.org/proposal/tzip-12

RG5 : Tous les billets mintés et associés à l'organisateur du concert sont listés : c'est à dire qu'ils peuvent être achetés.

RG6 : Après le withdraw du compte , il n'est plus possible d'annuler un concert

RG7 : En cas d'annulation, les billets non vendus sont burn et les acheteurs sont remboursés. Les acheteurs conservent néanmoins leur NFT. Celui ci pourra éventuellement servir à des réductions en cas de report.

RG8 : Les métadonnées du contrat doivent être stockées sur IPFS, et respecter le standard TZIP16
https://tzip.tezosagora.org/proposal/tzip-16/
Le standard TZIP12, indique comment utiliser les métadonnées dans le cadre d'un token FA2
Le standard TZIP21, défini le schéma JSON des metadonnées https://tzip.tezosagora.org/proposal/tzip-21/

d. Limites pour le POC du logiciel

Dans le cadre du POC, seul l'administrateur peut lister des billets à vendre. Le listing en second marché sera ajouté dans le MVP, pour le POC, les acheteurs de billets peuvent seulement les donner.

La gestion des reports et éventuelles actions commerciales associées ne font pas parti du POC.

Le NFT de billet ne contient pas d'image dans le POC, pour le MVP on associera l'affiche du concert.

e. Technologies retenues

i. Démarche

L'application est un ou une collection de smart contracts sur la blockchain Tezos.
On utilisera le réseau Ghostnet TESTNET pour le POC.

ii. Framework et langage

L'application doit être réalisée en JsLIGO

Il est conseillé de s'appuyer sur la librairie @ligo/fa https://packages.ligolang.org/package/@ligo/fa pour créer le projet.

iii. Sandboxing

Il est possible d'utiliser la solution flextesa https://gitlab.com/tezos/flextesa pour faire des tests avec une chaine exécutée localement.

iv. Utilitaires

La librairie taquito https://tezostaquito.io/ peut être utilisée pour créer les scripts de déploiement.

La CLI Taqueria https://taqueria.io/ peut être utilisée pour simplifier la gestion des différents outils, à défaut un Makefile ou package.json devra être fournit avec les différents scripts nécessaires à l'utilisation de la solution.
Il est nécessaire d'avoir Docker installé pour utiliser Taqueria.

3) Application Web utilisateur

a. Description

Cette application a pour objectif de proposer une UI web pour le service Justicket. Il s'agit d'une application web.

La réalisation de ce lot est confié par le client à un prestataire.

b. Stories

Story 1 : En tant qu'**Utilisateur**, je peux **me connecter avec mon wallet**

Story 2 : En tant qu'**Administrateur**, je peux **créer un concert**, afin de **que les NFT des billets soient listés**

Story 3 : En tant qu'**Administrateur**, je peux **déclarer un concert terminé**, afin d'**être payé**

Story 4 : En tant qu'**Utilisateur**, je peux **visualiser les concerts listés**, afin de **sélectionner et visualiser le détail d'un concert**

Story 5 : En tant qu'**Acheteur**, je peux **acheter un billet NFT**, afin de **pouvoir me rendre au concert**

Story 6 : En tant qu'**Acheteur**, je peux **voir mes billets NFT,** afin de **visualiser mes futurs concerts**

Story 7 : En tant qu'**Acheteur**, je peux **burn un de mes billets NFT,** afin de **signifier mon entrée au concert**

Story 6 : En tant qu'**Acheteur**, je peux **transférer un de mes billets NFT,** afin de **le donner à une autre personne.**

c. Règles de gestion

RG1 : Tous wallet Beacon doit être compatible avec l'application. La classe BeaconWallet de la librairie Taquito permet cette gestion.

RG2 : Les transferts s'effectuent vers une adresse Tezos valide

d. Limites pour le POC du logiciel

Bien que présent dans le POC d'API, la mise à jour des concerts ne fait pas partie du périmètre du POC application Web.

e. Technologies retenues

i. Démarche

L'application est une application frontend en ReactJS.

ii. Framework et langage

L'application doit être réalisée en Typescript et ReactJS

Le flux de données de l'application doit être géré avec Redux et Redux Loop

iii. Utilitaires

La librairie taquito https://tezostaquito.io/ peut être utilisée pour gérer la connexion du wallet ou soumettre des opérations.

L'indexer public tzkt.io peut être utilisé pour lire des données de la chaine, soit directement par API, soit via le SDK Typescript, soit en tant que plugin Taquito

iv. URIs

URL du noeud RPC Tezos : https://ghostnet.tezos.marigold.dev *nécessaire pour configurer le TezosTookit de taquito*

Documentation d'API Tezos : https://api.marigold.dev/general/ghostnet *à priori vous ne devriez pas avoir à vous en servir directement si vous utiliser Taquito ou Tzkt*

URL de l'API tzkt : https://api.ghostnet.tzkt.io

4) Application de Dashboard

a. Description

Cette application a pour objectif de fournir des graphiques de suivi à la DSI.

La réalisation de ce lot est confié par le client à sa Direction des Services Informatiques (DSI).

La DSI a également en charge de s'assurer que l'ensemble des projets s'intègrent et soit déployés pour le POC.

La DSI est libre de fixer l'infrastructure technique d'hébergement cloud du POC.

b. Stories

Story 1 : En tant qu'**utilisateur**, je peux **sélectionner un indicateur**, afin de **visualiser le graphique associé**

c. Indicateurs

IND1 : Nombre de billet NFT vendu par jour, sur un mois glissant

- Type: graphique à barre
- Axe X : Temps
- Axe Y : Nombre d'unités

IND2 : Valeur de billet vendu par jour, sur un mois glissant

- Type: graphique en ligne
- Axe X : Temps
- Axe Y : Valeur en tez

IND3 : Taux d'utilisation, pour chaque concert non annulé

- Type: graphique en camembert (un par concert)
- Secteurs : Billets non vendus, Billet vendus non utilisés, Billets utilisés

IND4 : Taux de billets transférés (indicateur global)

- Type: nombre
- Définition : pourcentage de billets transféré parmi l'ensemble des billets vendus

d. Limites pour le POC du logiciel

Bien que principalement destinés à la DSI, l'ensemble des graphiques sont publics.

Aucune gestion d'accès n'est donc à implémenter dans le cadre du POC.

e. Technologies retenues

i. Démarche

L'application est une application frontend en ReactJS.

ii. Framework et langage

L'application doit être réalisée en Typescript et ReactJS

Le flux de données de l'application doit être géré avec Redux et Redux Loop

Les graphiques peuvent être produit, au choix du prestataire, soit avec la librairie Rechart https://recharts.org , soit avec la librairie Victory https://formidable.com/open-source/victory/docs

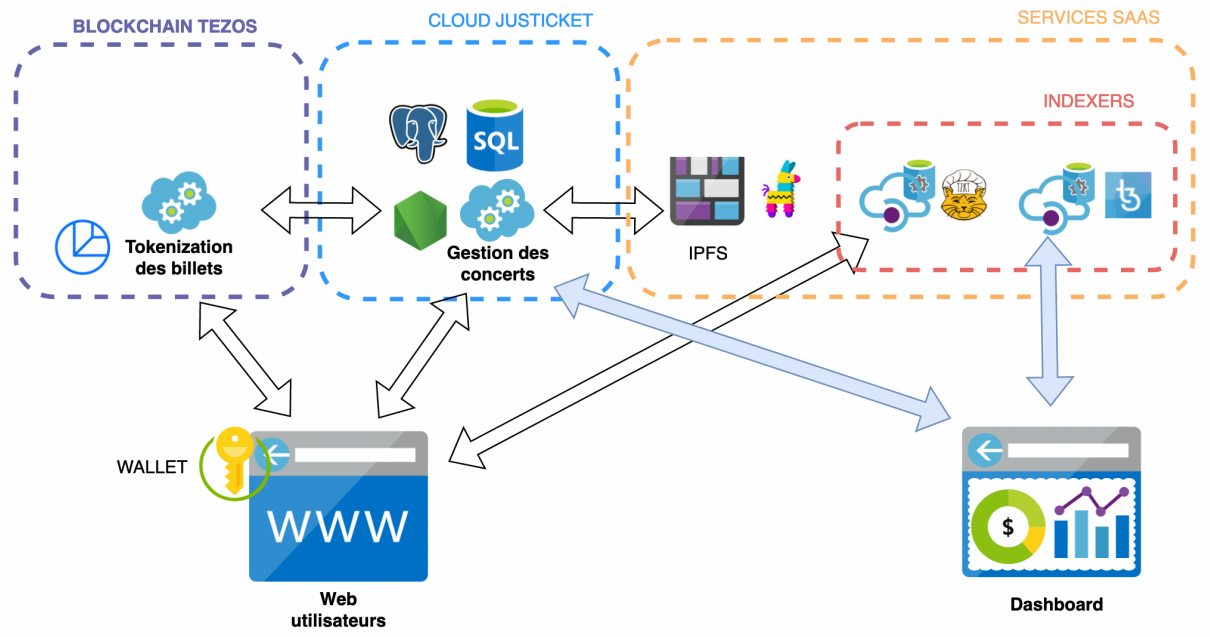
iii. Indexer

L'indexer public tzstats https://tzstats.com doit être utilisé pour récupérer les informations des contrats.

URL de l'API tzstats : https://api.ghost.tzstats.com

ARCHITECTURE

Diagramme des composants de la solution : SI de Justicket



Justicket

Glossaire blockchain

Tez : crypto-monnaie native de la blockchain Tezos. Son symbole est XTZ.

Mutez : 1/1000000ème (1 millionième) de tez. Utile pour manipuler des fractions de tez, car on ne manipule que des nombres entiers dans les smart contracts.

Mint : Créer un token.

Burn : Détruire un token, il peut s'agir d'une destruction logique, c'est à dire qu'on transfert le token à une adresse dont personne ne possède la clé privée, par exemple tz1burnburnburnburnburnburnburjAYjjX sur Tezos, ou 0x0000000000000000000000000000000000000000 sur Ethereum, sont souvent utilisées. Il est aussi possible de détruire physiquement un token en le supprimant du « storage » de son contrat. En revanche, les cryptomonnaies natives telle que le tez ou l'ether ne peuvent être que « burn » logiquement.

Smart contract : code executable au sein d'une blockchain. Un contrat se compose principalement de :

- Sa balance: un contrat peut recevoir ou envoyer des Tez
- Son storage: données pouvant être lues et écrites par le contrat
- Son code: composé d'un ensemble d'« entrypoints », des fonctions pouvant être appelées depuis l'extérieur de la chaine ou depuis un autre smart contract.

A la différence d'un code web classique un smart contrat doit être déterministe. Cela est nécessaire pour garantir le même état final après l'application d'une transaction, quel que soit le noeud qui produit le bloc.

Token : une représentation numérique d'un actif. Il s'agit simplement de données stockées au sein d'un smart contract

FT : token fongible. Permet de représenter des « objets » identiques en terme d'usage ou de valeur. Par exemple, 2 pièce de 1 Euro ou 2 XTZ

NFT : token non fongible. Permet de représenter des « objets » distincts en terme d'usage ou de valeur. Par exemple, 2 plaques d'immatriculation ou 2 billets d'un même concert, l'un pour la place 33 rang G et l'autre pour la place 13 rang B. Historiquement les NFT désignaient des tokens réellement uniques. De nos jours, le terme est couramment utilisé pour désigner des tokens avec un stock prédéfini et limité. La distinction entre FT et NFT devient poreuse avec les standards multi-assets tel que FA2.0 sur Tezos ou ERC1155 sur Ethereum.

Wallet : moyen de stockage de la clé privée d'une adresse sur une blockchain. Il peut s'agir d'un moyen non technologique - ***cold wallet*** - comme une feuille de papier; d'une application web ou mobile - ***hot wallet*** - comme Temple ou Airgap; ou d'un hardware spécifique - ***hardware wallet*** - comme Ledger. Les hot wallets facilitent l'utilisation d'applications, en conservant de manière sécurisée la clé privée dans une application. Ils peuvent parfois être couplés à un hardware wallet pour plus de sécurité, de la même manière qu'on peut utiliser un 2FA physique pour sécuriser des connexions.

Les tokens sont toujours stockés sur la blockchain, les wallets ne conservent que des secrets, jamais des actifs. Tant que vous conservez soit la pass phrase (liste de mots clés), soit la clé privée, en sécurité, vous pourrez toujours avoir accès à vos actifs, même si votre PC brule ou que le hardware wallet est détruit.

Node : serveur executant le programme de la blockchain. Il existe plusieurs type de noeud : des bakers qui produisent des blocs, des light clients qui contrôle des états récents, des full archive qui contiennent l'ensemble des données depuis l'origine (nécessaire pour les indexer),… Certains node peuvent exposer des RPCs afin de permettre à des applications de communiquer avec la chaine.

Indexer : Logiciel externe à la chaine qui indexe les données de la chaine afin de proposer un modèle de données exploitables, performant en lecture, souvent complété de dimensions pour faciliter la création d'applications web ou analytique. Il existe des indexer public - comme tzkt ou tzstats - et des solutions pour créer des indexer privés - comme DipDup, QuePasa ou Dappetizer. Les indexers public sont utiles au grand publique ou pour des MVP d'application. Les applications en production utilisent en général leur propre indexer privé, pour des questions de performance et de confiance.

CHARTE GRAPHIQUE

Police Logo et Titres : Fugaz One https://fonts.google.com/specimen/Fugaz+One

Police Textes : Source Sans Pro https://fonts.google.com/specimen/Source+Sans+Pro

Iconographie : Circum icons https://github.com/Klarr-Agency/Circum-icons

Logo

Couleur background : #FFFFFF

Couleur texte principal : #000000

Couleur 1 : #EA4949

Couleur 2 : #0C296B