Story Builder

Un constructeur d'*aventure dont vous êtes le héros*

Projet réalisé par

Jérémie LEYMARIE

Guillaume Thiebaud

Nabi Aries

Yanis Gherdane

Rihab Ammar

Introduction

Résumé

Story-Builder est une plateforme de création de jeu type *Histoire dont vous êtes le héros.* Le projet se décompose en deux parties : d'un côté la partie constructeur d'histoire, le *builder* et de l'autre le jeu en lui-même. L'idée générale est de permettre à la fois l'élaboration intégrale d'une histoire, la publication de celle-ci sur un store interne donnant accès aux joueurs à une bibliothèque communautaire.

Objectifs Techniques

Ce projet repose également sur deux ambitions techniques. D'une part, proposer un produit cross-platform : Mobile (IOS/Android), Web & Desktop, par le biais d'une *Progressive Web App (PWA).* D'autre part, le projet est pensé *offline-first*, c'est-à-dire que la connexion Internet ne doit être nécessaire que dans de rares cas (téléchargement d'histoires depuis le store, synchronisation entre appareils). L'essentiel de l'application doit pouvoir fonctionner entièrement en local.

Architecture technique du projet

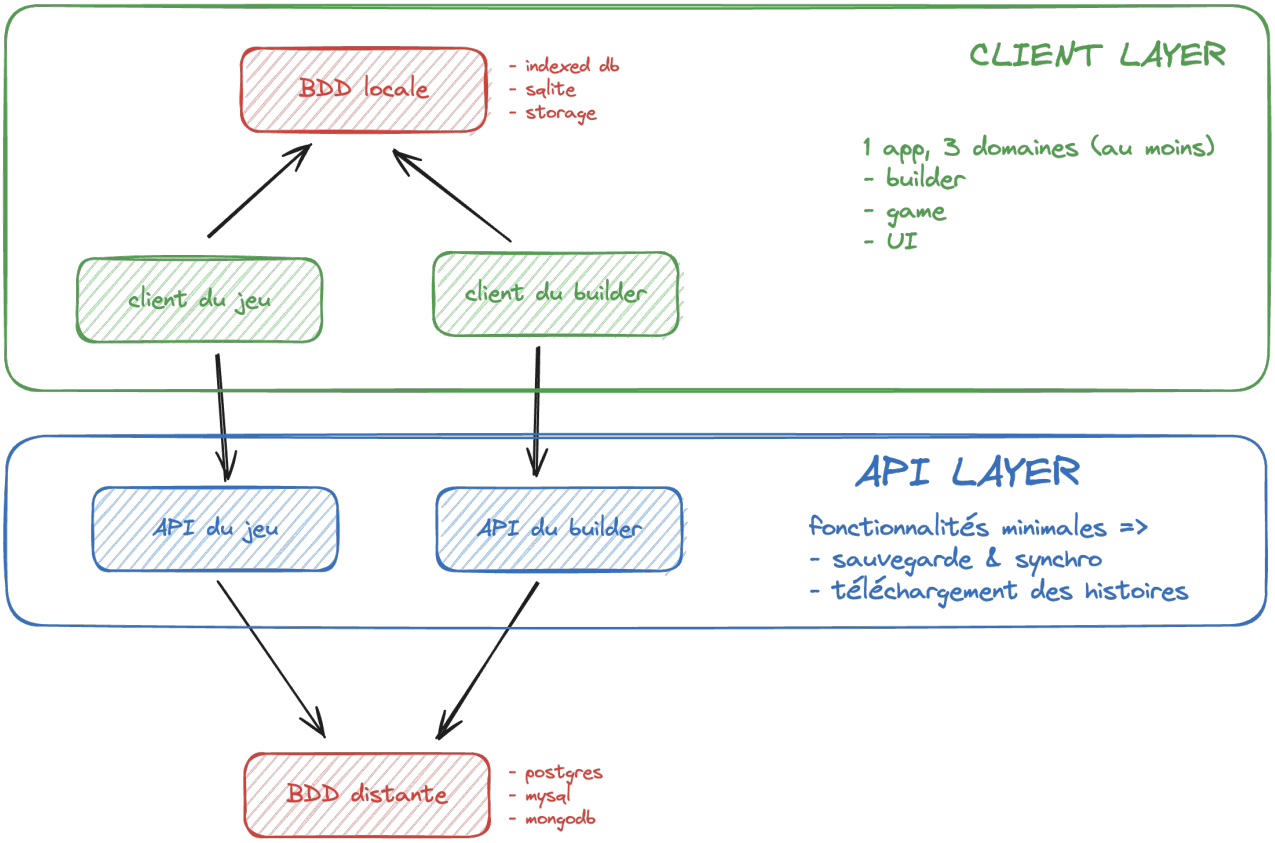

Fig 1. Schéma d'ensemble de Story Builder

Stack technique

Frontend

Le frontend est une application *React*, orchestrée par *Vite* et écrite en *Typescript*. La partie PWA est gérée par le plugin *vite-pwa.*
Quelques outils sont structurels pour le frontend :

- *Tanstack Router* gère le routing dans le frontend
- *TailwindCSS* est le moteur de style
- *ShadCN UI* est la librairie de composants graphiques qui forme la base de notre design-system
- *React Flow* permet de représenter des données sous forme de noeuds et de liens

Backend

Le backend est écrit en *Python*, en utilisant le framework web *Fast API.* La validation des requêtes et des réponses se fait via *Pydantic.* Le backend est typé via le mécanisme de type hints en python et la vérification statique des types se fait par *mypy.*

Bases de Données

Le project repose sur une base de donnée locale dans le navigateur, *IndexedDB.* On utilise un wrapper autour de l'API native, la libraire *Dexie.*

La base de donnée à distance est une base de donnée *MongoDB*, et les interactions avec elle se font par le driver python *pymongo.*

Liste de fonctionnalités

Builder

Page d'accueil

Owner : **Jérémie**
Tags : Front
Description :
Doit servir de Hub central à l'application. Donne accès à : builder, bibliothèque, parties en cours, partie actuelle.
Etat : En cours

Initialiser la gestion du stockage local

Owner : **Jérémie & Guillaume**
Tags : Front
Description :
- Initialiser l'instance Dexie
- Créer les modèles de données (v1)
- Exposer un repository pour s'interfacer avec la base de données
- Ecrire un premier exemple d'utilisation

Etat : Fini

Initialiser la gestion du stockage à distance

Owner : **Guillaume**
Tags : Back, Devops
Description :
- Dockerizer MongoDB pour faciliter l'utilisation à toute la team
- Initialiser le client Mongo dans l'app FastAPI
- Ecrire un premier exemple d'utilisation
- Avoir des petits helpers réutilisables liés à MongoDB (formatage des ids)

Etat : Fini

Authentification (Back)

Owner : **Guillaume**
Tags : Back
Description :
- Route de connexion
- Route d'inscription
- Création/Vérification de token JWT
- Hashing de mot de passe
- Validation via pydantic

Etat : Fini

Page de connexion (Front)

Owner : **Nabi**
Tags : Front
Description :
- Création de la page */signin* et */signup*
- Formulairesavec validation utilisant *zod, react-hook-form* et le *design-system*
- Interaction avec l'API pour valider les actions
- Redirection vers la page d'accueil en cas de succès, affichage d'un message d'erreur sinon

Etat : En cours

Page d'inscription (Front)

Owner : **Nabi**
Tags : Front
Description :
- Création de la page */signup*
- Formulaire avec validation utilisant *zod, react-hook-form* et le *design-system*
- Interaction avec l'API pour valider les actions
- Redirection vers la page d'accueil en cas de succès, affichage d'un message d'erreur sinon

Etat : En cours

Page de liste des histoires (Builder)

Owner : **Jérémie**
Tags : Front
Description : La page doit afficher la liste de toutes les histoires crées par un utilisateur.
- Récupère la liste des histoires depuis la base de données locale
- Affiche des cartes représentant les histoires d'un utilisateur
- Au clic, amène sur l'interface du Builder correspondant à l'histoire

Etat : Fini

Création d'une nouvelle histoire

Owner : **Jérémie**
Tags : Front
Description :
- Dans la liste des histoires, la première carte de la liste propose de créer une histoire
- Formulaire dans une modale, avec validation et messages d'erreur
- Si la création est fructueuse, l'histoire est stockée dans la base de données locale
- Après validation de l'action, redirection vers le builder dans l'histoire nouvellement créée.

Etat : Fini

Affichage des scènes dans l'espace de création

Owner : **Jérémie**
Tags : Front
Description :
- Dans le builder, les scènes enregistrées doivent être affichées
- Leurs données sont chargées depuis la base de données locale
- Leur position est enregistrée sauvegardée
- On peut interagir avec les noeuds, les relier, les déplacer

Etat : Fini

Création/Modification d'une scène dans une histoire

Owner : **Jérémie**
Tags : Front
Description :
- Dans l'interface du builder, une toolbar à gauche permet de créer une scène
- Une modale s'ouvre, avec un formulaire validé par zod, permettant de créer une scène
- Dans sa v1, permet d'ajouter un titre et un contenu
- Au succès, la modale se ferme et le nouveau contenu est enregistré dans la base de données locale
- L'interface est mise à jour avec les nouvelles informations
- Quand une scène est déplacée dans l'interface, sa nouvelle position est enregistrée dans la base de données locale

Etat : Fini

Espace *Whiteboard/Croquis*

Owner : **Guillaume**
Tags : Front
Description :
- Une page (v1, peut-être intégrée dans l'interface du Builder plus tard) doit donner à un espace whiteboard
- Sur cet espace, on a accès à l'outil *Excalidraw* qui permet de dessiner, faire des croquis à main levée et des schémas

Etat : Fini

Ajouter/Modifier/Supprimer des boutons dans une scène

Owner : **Jérémie**
Tags : Front
Description :
- Le formulaire d'édition et de création d'une scène doit permettre d'ajouter/retirer/modifier des boutons
- Les boutons doivent s'afficher dans le builder

- Ils doivent être enregistrés dans la base de données locale

Etat : Fini

Lier des boutons à des scènes

Owner : **Jérémie**
Tags : Front
Description :
- Dans le builder, il doit être possible de relier des boutons à une scène
- Un bouton ne peut pas se lier à sa propre scène
- Les liens doivent être enregistrés automatiquement dans la base de données locale

Etat : En cours

Noeud de départ d'une histoire

Owner : **Non assigné**
Tags : Front
Description :
- Lors de la création d'une histoire, une noeud de départ doit être créé dans la base de données locale
- Le Builder doit afficher ce noeud de manière distinctive
- Il doit être possible de lier de noeud à une scène, mais rien ne peut être lié à ce noeud

Etat : Pas commencé

Bouton de synchronisation à distance

Owner : **Non assigné**
Tags : Back, Front
Description :
- Dans le builder, un bouton doit permettre de synchroniser l'histoire dans la base de données distante (ça demande de refactoriser la navbar)
- Une route API doit permettre de synchroniser l'histoire entre l'état de la base de données locale et la base de données distante

Etat : Pas commencé

Publication d'une histoire

Owner : **Non assigné**
Tags : Back, Front
Description :
- Dans le builder, un bouton doit permettre de publier l'histoire => passage de statut *draft* à status *published*
- La modification s'effectue d'abord dans la base de données distante via API, puis dans la base de données locale

Etat : Pas commencé

Game

Page du store d'histoires

Owner : **Guillaume**
Tags : Front
Description :
- Une page, accessible depuis la page d'accueil donne accès à un store d'histoires
- Sur cette page, on affiche une liste de cartes, représentant chacune une histoire

Etat : En cours

Route API pour le store

Owner : **Non assigné**
Tags : Back, Front
Description :
- Une route API permet de récupérer toutes les histories, dont le statut est *published*
- Seulement les informations essentielles doivent être récupérées (pas besoin des scènes)

Etat : Pas commencé

Télécharger une histoire (BACK)

Owner : **Non assigné**
Tags : Back
Description :
L'application fonctionnant essentiellement hors-ligne, le téléchargement d'une histoire est un pré-requis pour pouvoir jouer une partie.
Une route API doit permettre d'envoyer tout le contenu d'une histoire.
Points exploratoires à débroussailler :
- Le téléchargement peut être long, comment Fast API gère ce type de problématique (voir HTTP Polling)

Etat : Pas commencé

Télécharger une histoire (FRONT)

Owner : **Non assigné**
Tags : Front
Description :
L'application fonctionnant essentiellement hors-ligne, le téléchargement d'une histoire est un pré-requis pour pouvoir jouer une partie.
- Au clic sur une carte dans le store, une modale propose de lancer le téléchargement de l'histoire ou d'annuler.
- Le clic sur *Cancel* referme la modale

- Le clic sur *Download* lance le téléchargement via l'API
- La UI doit refléter la progression du téléchargement de l'histoire
- Une fois les données de l'histoire reçues, elles doivent être enregistrées dans la base de donnée locale.
- Après le téléchargement, l'interface propose de lancer la partie ou de revenir sur le store.
- Point de vigilance sur la lenteur du call API dans le cas d'une longue histoire => gérer le cas (HTTP polling, batching, à explorer)

Etat : Pas commencé

Page *Parties en cours*

Owner : **Non assigné**
Tags : Front
Description : La page doit afficher la liste de toutes les histoires téléchargées par un utilisateur.

- Récupère la liste des histoires depuis la base de données locale
- Affiche des cartes représentant les histoires téléchargées d'un utilisateur
- Au clic, redirige vers la page de détail d'une partie
- La page doit être pensée d'abord pour mobile, mais doit être 100% responsive

Etat : Pas commencé

Page de détail d'une partie

Owner : **Non assigné**
Tags : Front
Description : La page doit détailler les informations d'une partie.

- On peut y lire des informations sur l'histoire : titre, résumé
- On affiche l'image de couverture de l'histoire (background)
- La page doit être pensée d'abord pour mobile, mais doit être 100% responsive
- On y trouve également l'endroit où le joueur s'est arrêté (au moins le titre de la scène)
- On peut imaginer d'autres infos (TDB), comme le pourcentage de progression dans l'histoire, les stats du personnage au moment où iel s'est arrêté.e, etc…
- Un bouton "Jouer" permet de rediriger vers la page de jeu, à l'endroit où le.a joueur.euse s'est arrêté.e
- Charge les informations depuis la base de données locale

Etat : Pas commencé

Page de jeu

Owner : **Non assigné**
Tags : Front
Description :

- C'est l'élément central du jeu : la page que le.a joueur.euse lit et avec laquelle iel interagit pour se déplacer dans l'histoire
- Elle affiche le titre de la scène, le contenu de la scène

- Si la scène a une image, elle est présente
- Affiche les boutons pour se déplacer vers une autre scène
- Au clic sur un bouton, on redirige vers la scène correspondante, et on met à jour l'état actuel de la partie dans la base de données locale
- La page doit être pensée d'abord pour mobile, mais doit être 100% responsive
- Charge les informations depuis la base de données locale

Etat : Pas commencé

Pouvoir finir une histoire

Owner : **Non assigné**
Tags : Front
Description :

- Le jeu doit être capable de gérer une fin d'histoire, c'est-à-dire une scène qui ne contient aucune action
- La page de jeu doit afficher le texte de fin et proposer de revenir à l'écran principal.
- On enregistre dans la base de données locale que le.a joueur.euse a fini l'histoire.

Etat : Pas commencé

Technique

Typecheck du code python

Owner : **Jérémie**
Tags : Back, DevOps
Description :

- Ajouter mypy (static type checker) au projet
- Créer un script bash qui type check l'intégralité de la codebase
- Ajouter un hook de pre-commit qui vérifie le typage
- Ajouter une CI qui vérifie le typage (BONUS)

Etat : En cours

Ajouter des règles de linting/formatage proches des défauts de l'industrie

Owner : **Non assigné**
Tags : Front, DevOps
Description :

- Ajouter plus de règles eslint (imports, unused-vars, etc…)
- Forcer l'application des règles dans un hook de pre-commit
- Ajouter le check en CI

Etat : Pas commencé

BONUS

A détailler

Rendre la toolbar rétrécissable
Créer un personnage dans l'histoire (BUILDER & GAME)
Avec des inputs utilisateurs (nom, etc…)
Ajouter des statistiques de personnages (BUILDER & GAME)
Par ex => PV, Intelligence, Charisme, etc…
Utiliser des variables dans le texte de l'histoire (BUILDER & GAME)
Gestion de la musique (BUILDER & GAME)
Connexion par SSO
Upload de fichiers
Gestion d'inventaire
Combat
Conditionner l'histoire en fonction de choix passées
Pouvoir personnaliser les couleurs
Pouvoir personnaliser les boutons
Pouvoir personnaliser la police
Ajouter un wiki à une histoire
Intégrer une documentation utilisateur in-app
Export JSON d'une histoire