YetiScape

Le nouveau Escape Game coopératif totalement en ligne !

Alexandre BIAUD, Raoul EL MIR, Ambre HUTIER, Lilia KAROUIA, Swan Maillard

Table des matières

Discussion du projet

Idée initiale

Escape Game ludique pour Collégiens

Transformé

Escape Game coopératif multijoueur

Objectifs du projet

En Ligne

Un jeu totalement dématérialisé

Multijoueur

Un jeu à faire à plusieurs pour pouvoir réussir

JS/HTML/CSS

Montée en compétence :

- Code
- Site web from scratch

Base de Données

Communication & sauvegarde des données

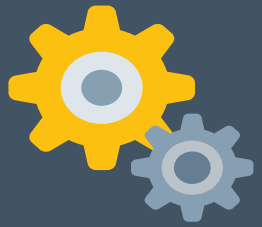
Mise en place

Besoins et Séparation des tâches

Front

Scenario du jeux
Gestion du front end

Back



Création d'un serveur
Gestion du back end : serveur, sockets, …

Chat

Création d'un chat écrit et vocal

Prévision du projet

Lundi	Mardi	Jeudi	Vendredi	Lundi
29	30	2	3	6
Scénario terminé		Mise en commun front/back	Mise en commun back/chat	
BD créée				

Architecture

Scénario du jeu

Architecture global du projet

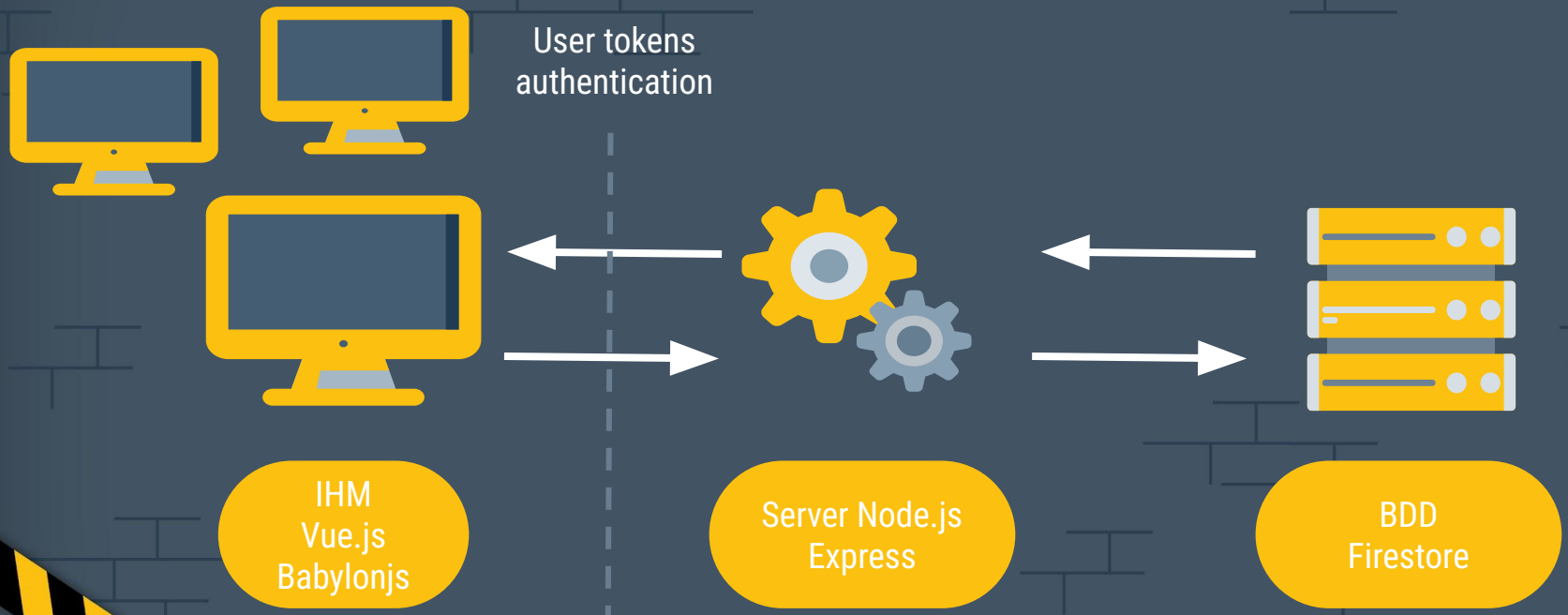
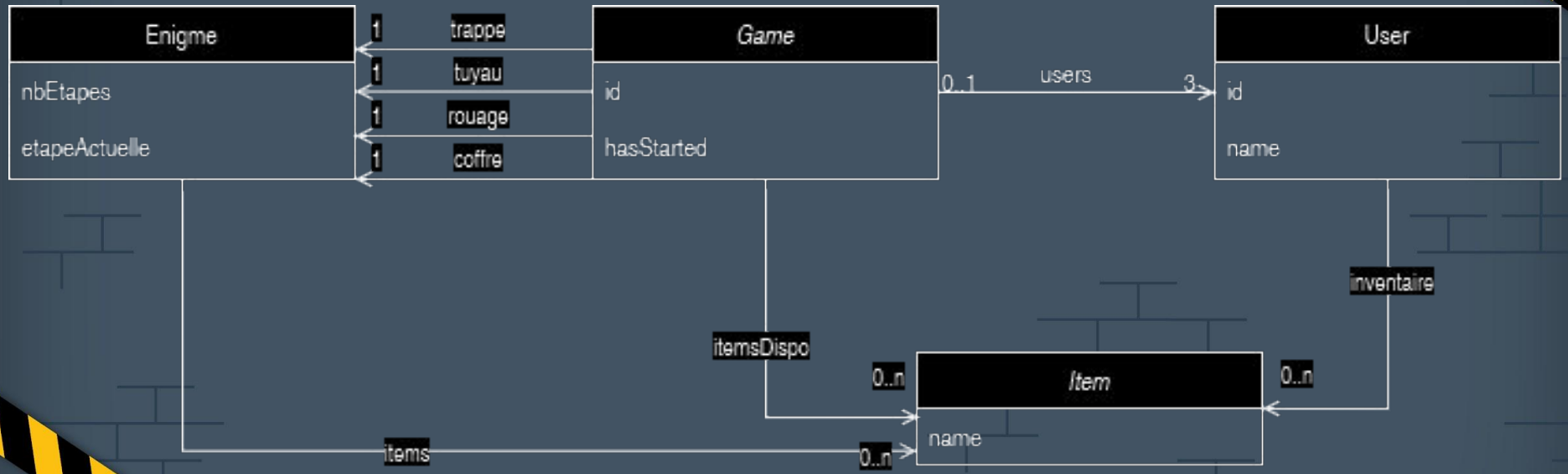
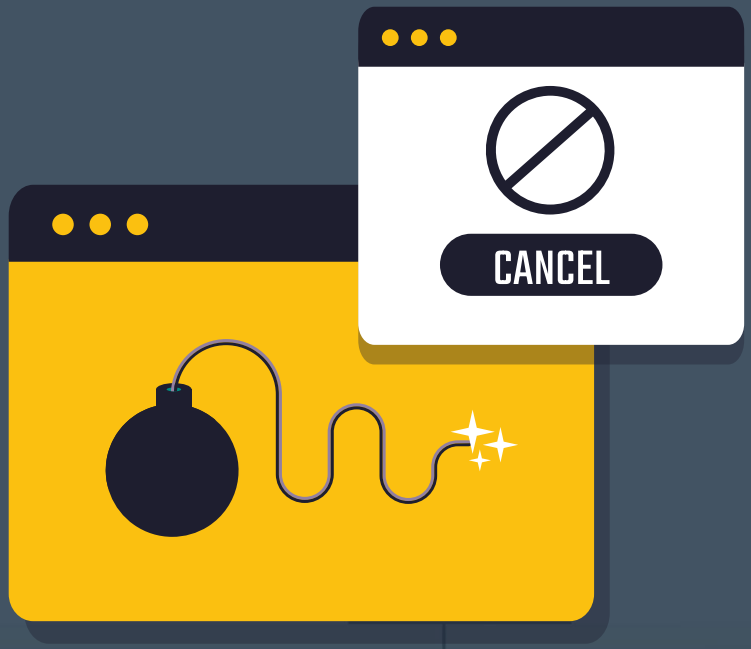


Diagramme de classes



Front-end

Technologies



Framework Vue.js

Qu'est-ce que c'est ?

- S'appuie sur les standard JavaScript, HTML et CSS
- Système réactif et librairie simple d'utilisation

Pourquoi nous l'utilisons ?

- Simplifier le code
- Créer une architecture de routage et de pages rapidement
- Découvrir un nouveau framework présent en entreprises
- Intégrer des modules comme socket.io

Babylonjs

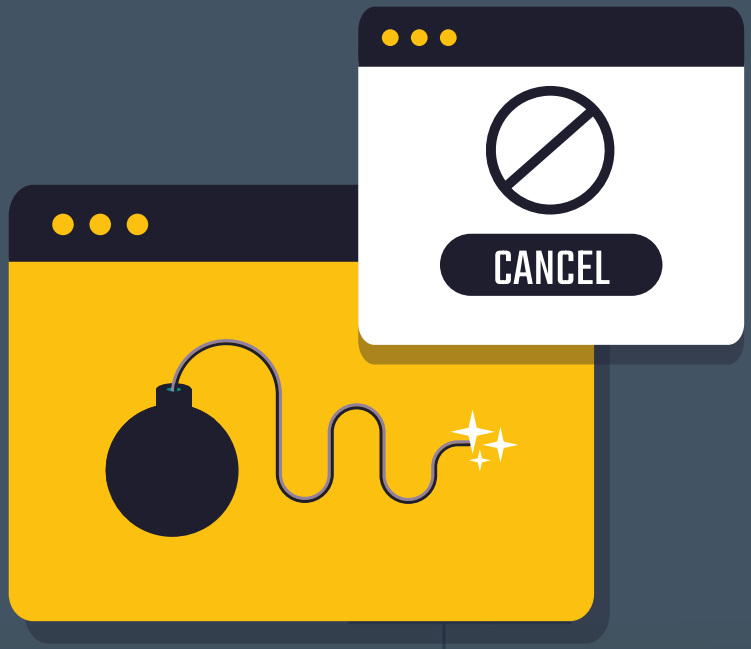
Modélisation de scènes 3D

Modélisation des salles des joueurs

- Rapide
- Intégrable sur Vue js
- Base solide

Chat

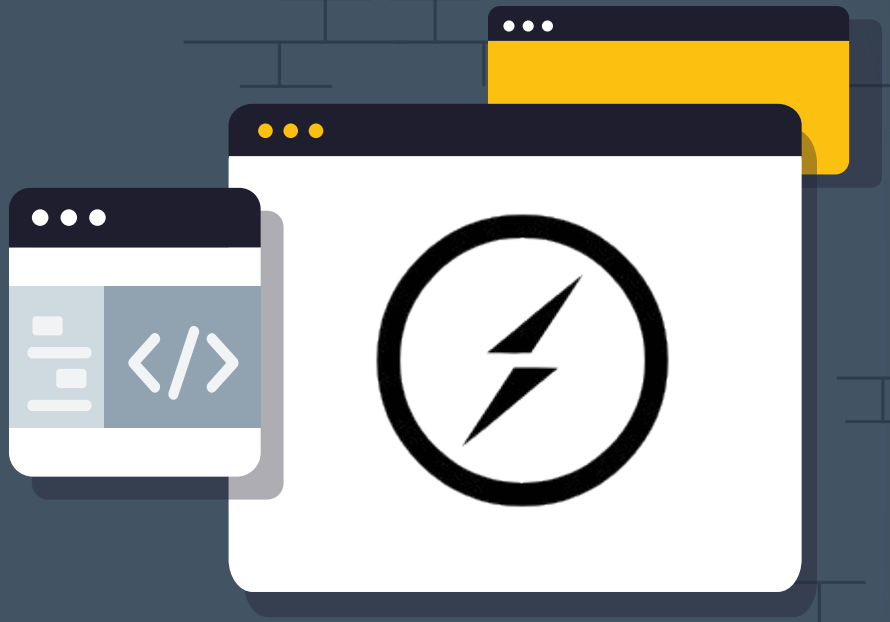
Architecture



Chat écrit

Socket.IO

- Communication en temps réel
- Basée sur des événements
- TCP
- Développement rapide et simple
- Léger



Chat vocal

WebRTC

- Real-Time-Communication

PeerJS

- Enveloppe WebRTC
- Complete, configurable
- Uniquement avec les IDs

Back-end

Architecture

Node.js - API Express

Qu'est-ce que c'est ?

- Node.js : Plateforme js côté serveur
- Express : Framework web facilitant la création d'API

Pourquoi nous l'utilisons ?

- Efficacité de mise en place
- Création d'une API pour orchestrer les différentes parties du jeu

Architecture de l'API

Routeur

Redirige les requêtes HTTP vers les contrôleurs.

Contrôleurs

Effectuent les actions liées à chaque requête. Communiquent avec les services

Services

Communiquent avec la base de données.

Modèles

Définissent les différentes entités de l'application.

Bases de données

Firestore

- Service en ligne de base NoSQL fournit par Google
- Limite d'accès lecture/écriture atteinte

SQLite

- Base NoSQL dans un fichier local
- Facile à implémenter et utiliser

Modèles

User

id: number
name: string
gameId: number
salle: number
items: Item[]

Game

id: number
users: User[]
itemsDispo: Item[]
enigmes: Enigme[]
dateStart: number
callId: number

Routeur HTTP

/...

POST /create {username}

POST /join {username, gameId}

GET /waiting-room/:gameId

TOKEN JWT
userId

/game/...

GET /salle

POST /pick-item {item}

POST /coffre/solve {code}

...

Sockets

chat/

/user-join

/message

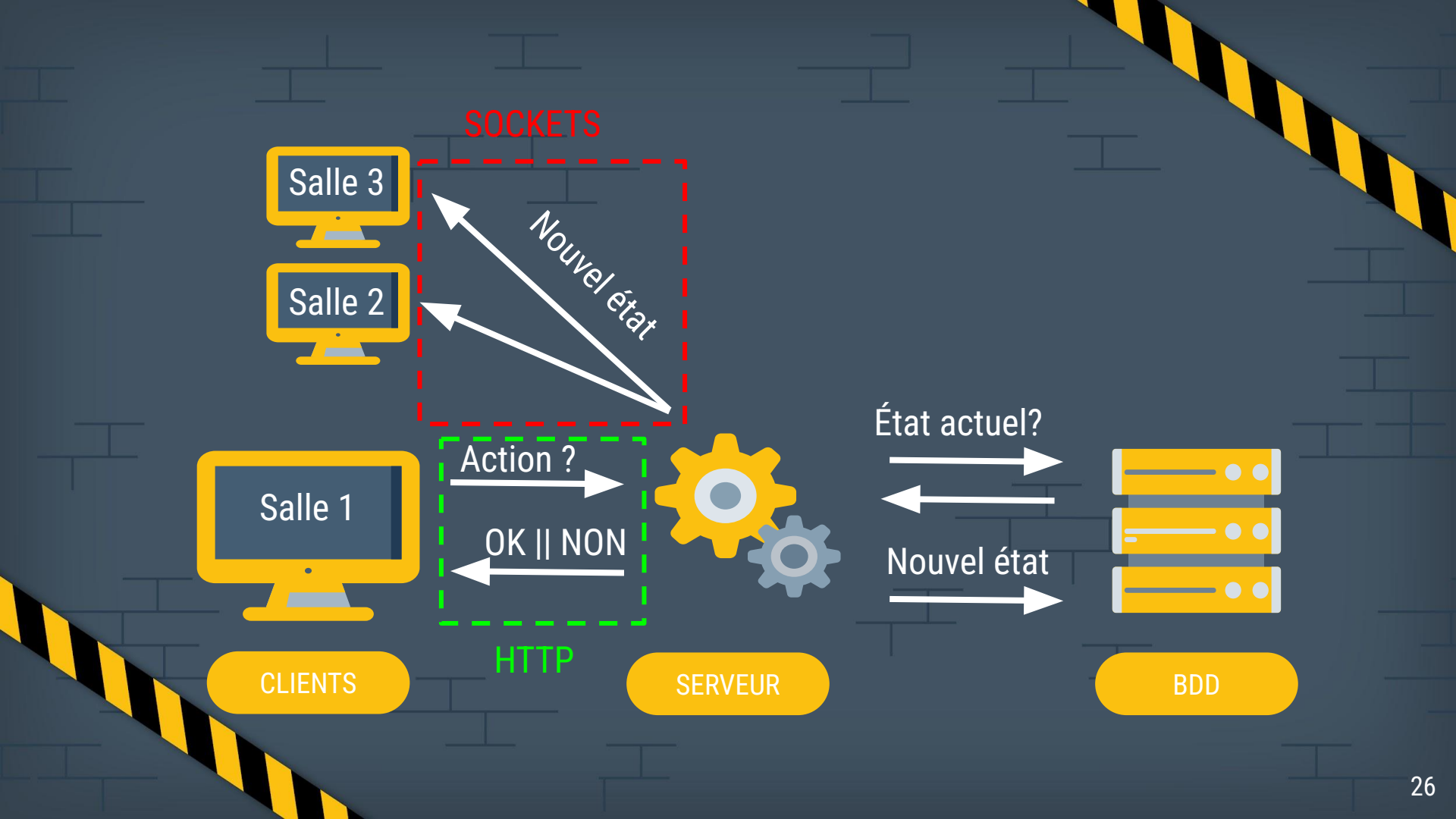
waiting-room/

/new-user

/start-game

game/

/tuyau-sent
/trappe-opened
/trappe-item-added
/trappe-item-removed
/doors-opened


SOCKETS
Salle 3
Salle 2
Nouvel état
Salle 1
Action ?
OK || NON
HTTP
CLIENTS
SERVEUR
État actuel?
Nouvel état
BDD