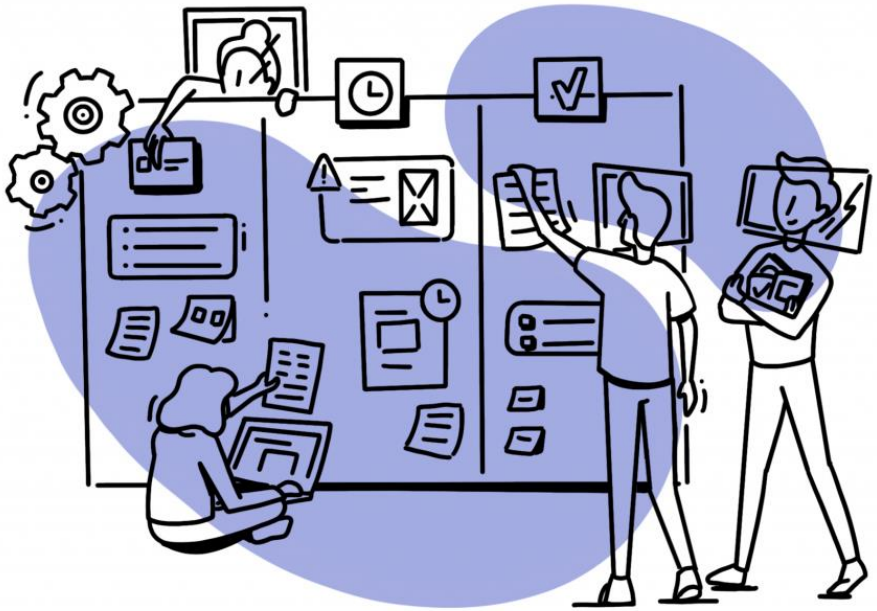

Projet GED'IMAGINATION
Approche agile / Security By Design

BTS SIO2 SLAM

Backlog de produit *Ged'Imagination*

	N°	En tant que ...	je souhaite ...	Priorité
Application web [AppWeb]	1	gestionnaire	paramétrer les données du jeu-concours	**Basse**
	2	visiteur	m'inscrire pour participer au jeu-concours	**Moyenne**
	3	utilisateur	me connecter	**Moyenne**
	4	participant	poster ma réalisation	**Haute**
	5	gestionnaire	générer et publier le classement du jeu concours	**Haute**
	6	visiteur	visualiser le classement du jeu concours	**Haute**
	7	participant	voir le nombre de Gaimes obtenu par ma réalisation	**Moyenne**
Application mobile [AppMobile]	8	gestionnaire	importer les données sur les réalisations des participants	**Haute**
	9	votant	voter pour mes réalisations préférées	**Haute**
	10	gestionnaire	exporter les données sur les votes	**Haute**

Sprint 2

Référence / Titre	Scénarios / Critères d'acceptation
[AppMobile] **User Story N°8** **En tant que** gestionnaire, **je souhaite** importer les données sur les réalisations des participants	**Etant donné** que je suis connecté à l'application mobile, **lorsque** je demande l'importation **alors** les données sont enregistrées dans la BDD embarquée ✓ : l'importation est confirmée ✘ : la date de fin des inscriptions n'est pas atteinte, je ne peux pas demander l'importation ✘ : les votes ont commencé, je ne peux pas demander l'importation

Evil User Stories	Security User Story - ***Mesures de sécurité à mettre en place***
En tant qu'attaquant, **je veux** effectuer une élévation de privilèges pour accéder à des fonctionnalités et/ou données que je ne suis pas autorisé à voir	**En tant que** développeur, **je veux** m'assurer que les utilisateurs ne peuvent accéder qu'aux fonctionnalités/données qui correspondent à leurs privilèges • *Mettre en place un système d'authentification et d'autorisation pour l'utilisation de l'API* • *N'exposer que les données/ressources nécessaires*
En tant qu'attaquant, **je souhaite** obtenir des informations techniques qui pourraient m'aider à comprendre le fonctionnement de l'application/du serveur en provoquant volontairement des erreurs	**En tant que** développeur, je veux m'assurer que tous les messages d'erreurs sont génériques • *Côté applicatif, gérer les comportements inattendus (gestion des exceptions) et définir des messages d'erreurs génériques - Ne pas afficher la trace détaillée de l'erreur (stacktrace)* • *Côté serveur, générer des messages d'erreurs standards (404, 500,…) ne donnant aucune information technique (version serveur web, version des composants…)*

Sprint 2

- User Story N°8 - Découpage en tâches
  - Créer la base de données embarquée [table(s) utile(s)]
  - Maquetter et créer l'activité
  - Côté serveur, créer et tester le service web (API) – Mettre en place un système d'authentification et d'autorisation
  - Coder le traitement d'import des données
  - Préparer la démo
  - Actualiser la documentation technique

Sécurité API – Principes de sécurité applicative

- A lire
  - https://blog.octo.com/securiser-une-api-rest-tout-ce-quil-faut-savoir/
  - https://www.silicon.fr/avis-expert/securite-des-api-les-meilleures-pratiques-pour-securiser-les-donnees-et-linfrastructure#

Sécurité API – Principes de sécurité applicative

- ## Ressources publiques ou privées

  Toutes les ressources exposées par votre API ne nécessitent pas le même niveau de vigilance

  - **Ressources publiques** : les données exposées sont publiques, l'accès à l'API ne nécessite aucune connexion, la mise en place d'une clé d'API permet toutefois de gérer les quotas d'utilisation des ressources et ainsi éviter les abus

  - **Ressources privées** : seules certaines ressources sont exposées et leur accès est restreint / limité à des utilisateurs autorisés. L'API peut être consommée par un ou plusieurs clients, internes à l'entreprise ou externe.

Sécurité API – Principes de sécurité applicative

- Les 3 piliers de la sécurité applicative
  - **Authentification** : l'authentification, c'est savoir qui appelle notre API. Un ensemble d'informations doivent être fournies pour identifier l'utilisateur appelant
  - **Authorization** : une fois que la personne qui appelle votre API a été identifiée, l'autorisation consiste à savoir ce que cette personne a le droit de faire (l'appelant a-t-il le droit de consommer cette ressource ou non ?)
  - **Accountability** : Il s'agit de savoir qui a fait quoi, quand, avec quelles ressources, on parle de traçabilité.
    Un fichier de logs va enregistrer pour chaque appel HTTP les informations suivantes : *la ressource consommée, le verbe HTTP sur cette ressource, le code HTTP de la réponse, l'application cliente consommatrice de la ressource, l'utilisateur à l'origine de l'appel, la date/heure de l'appel*

Sécurité API

- Risques potentiels liés aux API

Type d'attaque	Mesures d'atténuation des risques
Une **injection** se produit lorsqu'un attaquant insère des commandes ou du code malveillants dans un programme, généralement à la place d'une entrée utilisateur ordinaire (comme un nom d'utilisateur ou un mot de passe). Les injections SQL sont un type spécifique d'attaque par injection qui permet à l'attaquant de prendre le contrôle d'une base de données SQL.	Validation et nettoyage de toutes les données dans les requêtes d'API. Restriction des données de réponse pour éviter les fuites involontaires de données sensibles.
Le **Cross-Site Scripting (XSS)** est un type d'attaque par injection au cours de laquelle un cybercriminel profite d'une vulnérabilité pour insérer un script malveillant (souvent en JavaScript) dans le code d'une application ou page Web.	Validation des données en entrée. Filtrage et échappement de caractères.

Sécurité API

- Risques potentiels liés aux API

Les attaques par **déni de service distribué (Distributed Denial-of-Service ou DDoS)** rendent un réseau, un système ou un site Web indisponible pour ses utilisateurs, le plus souvent en générant une quantité de trafic supérieure à celle qu'il peut prendre en charge. Les endpoints d'API s'ajoutent à la liste de plus en plus longue des cibles DDoS.	Limitation du trafic et de la taille de la charge utile.
Les attaques **Man-in-the-Middle (MitM)**, se produisent lorsqu'un cybercriminel intercepte le trafic entre deux systèmes communicants et se fait passer pour l'autre auprès de chacun d'eux, agissant ainsi comme un intermédiaire invisible. Avec les API, les attaques MitM peuvent survenir entre le client (application) et l'API, ou entre l'API et son endpoint.	Chiffrement du trafic en transit.
Le **credential stuffing** désigne l'utilisation d'informations d'identification volées sur les endpoints d'authentification des API pour obtenir un accès non autorisé.	Exploitation d'un flux de renseignements pour identifier le credential stuffing et définition de limites de trafic afin de contrôler les attaques par force brute.

Sécurité API – Principes de sécurité applicative

- Principes de sécurité à mettre en place
  - Utiliser une authentification et une autorisation fortes
  - Valider les données en entrée
  - Appliquer le principe de moindre privilège
  - Chiffrer le trafic (TLS)
  - Ne pas exposer plus de données que nécessaire
  - Limiter le trafic (quotas)