

# SoftDesk: mise en place d'une API REST sécurisée

Openclassrooms, parcours python, projet 10

Bérenger Ossété Gombé

8 décembre 2022

# Table des matières

- 1 Introduction
- 2 Exigences
- 3 API REST
- 4 Documentation
- 5 Sécurité
- 6 Démonstration
- 7 Développement
- 8 Conclusions

# Introduction

Bérenger Ossété Gombé

- Bac scientifique en 2013
- Maîtrise en informatique (génie logiciel) en 2017
- Reconversion web chez Openclassrooms depuis janvier 2022

# Créez une API sécurisée RESTful en utilisant Django REST

## Projet 10 : parcours python

- Documenter une application
- Créer une API RESTful
- Sécuriser une API (OWASP et RGPD)

# Contexte du projet



## SoftDesk

- Société d'édition de logiciels.
- Veut développer un **système de suivi de problèmes**.
- Nous sommes ingénieur *backend* sur cette application.

# Exigences

## Exigences fonctionnelles

- En tant qu'utilisateur nous pouvons **nous authentifier**.
- En tant qu'utilisateur nous pouvons **manipuler des projets**.
- En tant que contributeur d'un projet nous pouvons **manipuler les problèmes** qui lui sont liés.
- En tant que contributeur d'un projet nous pouvons **commenter les problèmes** qui lui sont liés.
- En tant qu'auteur d'un problème, d'un projet, ou d'un commentaire nous pouvons **l'actualiser et le supprimer**.
- En tant qu'auteur d'un projet nous pouvons **ajouter ou supprimer un collaborateur**.

# Différents droits d'accès

## Droits d'accès

	Auteur	Collaborateur	Utilisateur
créer un projet	✓	✓	✓
consulter un projet	✓	✓	-
éditer un projet	✓	-	-
supprimer un projet	✓	-	-
créer un problème	✓	✓	-
consulter un problème	✓	✓	-
éditer un problème	✓	-	-
supprimer un problème	✓	-	-
créer un commentaire	✓	✓	-
consulter un commentaire	✓	✓	-
éditer un commentaire	✓	-	-
supprimer un commentaire	✓	-	-

# API REST

## Les ressources

Nom	URI
Projets	<i>/projects/</i>
Collaborateurs	<i>/projects/{id}/users/</i>
Problèmes	<i>/projects/{id}/issues/</i>
Commentaires	<i>/projects/{id}/issues/{id}/comments/</i>

## Actions

Création	POST
Accès	GET
Mise à jour	PUT
Suppression	DELETE



# Documentation

# Pourquoi documenter ?

Plusieurs raisons de maintenir la documentation d'un projet

- TODO

# Quoi documenter ?

## Tout au long du cycle de vie du logiciel

- Les exigences (cahier des charges)
- Le développement (dans le code source et la documentation externe)
- Les tests (plan de test, stratégie de test, ...)
- Les analyses et audits
- Le manuel pour utilisateurs ou développeurs

# Documentation et développement

## *Documentation Driven Development*

- Mise en place d'une documentation **avant** l'implémentation.

## Documentation de l'API

- Documentation *via* un wiki directement depuis *github*.
- Utilisation de sphinx

# Documenter le code

## Sphinx

- Utilisée par read the docs
- Permet de créer une documentation à partir de fichiers reStructuredText.

# Documentation : Sphinx

This is a Title

That has a paragraph about a main subject and is set when the '=' is at least the same length of the title itself.

Subject Subtitle

Subtitles are set with '-' and are required to have the same length of the subtitle itself, just like titles.

Lists can be unnumbered like:

- \* Item Foo
- \* Item Bar

Or automatically numbered:

- #. Item 1
- #. Item 2

Inline Markup

Words can have *\*emphasis in italics\** or be **\*\*bold\*\*** and you can define code samples with back quotes, like when you talk about a command: `'sudo'` gives you super user powers!

Figure – Exemple de document rédigé *via* la syntaxe reStructuredText

# Sécurité

# Enjeux de la sécurité

Tout est une question de risques

Risque = un impact x probabilité d'occurrence

Rédaction d'un modèle de menaces (*threats modelling*)

Quatre questions à se poser<sup>1</sup> :

- Sur quoi travaillons-nous ?
- Qu'est-ce qu'il pourrait arriver de mauvais ?
- Quoi faire si cela arrive ?
- Avons-nous fait du bon travail ?



## Le top 10 des menaces de sécurité

- 1 TODO enumeration des menaces

# Sécurité mise en place par Django

TODO : qu'est-ce que django fait automatiquement ?

TODO : quoi configurer ?

TODO : comment configurer ?

# Renforcer la sécurité

## Sécurité supplémentaire

- Authentification par nom d'utilisateur et mot de passe :
  - Vérification de la sécurité du mot de passe (TODO : voir django password validator)
- Mise en place d'un système de rôles :
  - Administrateur
  - Auteur d'un projet, d'un problème ou d'un commentaire
  - Collaborateur
  - Utilisateur

# Tester la sécurité de l'API

## Différents types de tests

- *Scan* de vulnérabilités
- Tests de pénétrations
- Audits de sécurité
- ...

Comment tester la sécurité de l'API ?

→ *Via* les tests de Django

TODO : différents niveau de tests, ici on parle de tests non fonctionnels d'acceptations et non pas de tests fonctionnels unitaires (→ en complément, non pas à la place).

# Démonstration avec POSTMAN

## Démonstration

- 1 CRUD d'un utilisateur
- 2 Connexion
- 3 CRUD d'un projet
- 4 CRUD d'un collaborateur
- 5 CRUD d'un problème
- 6 CRUD d'un commentaire
- 7 Illustration des droits
- 8 Déconnexion

# Développement

# Vue d'ensemble

# Les rôles



# L'authentification

# Les projets

# Les problèmes

# Les commentaires

# Conclusions