SoftDesk: mise en place d'une API REST sécurisée

Openclassrooms, parcours python, projet 10

Bérenger Ossété Gombé

11 décembre 2022

Table des matières

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

Contexte du projet

SoftDesk

- Société d'édition de logiciels.
- Veut développez 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	/*projects*/
Collaborateurs	/*projects*/{*id*}/*users*/
Problèmes	/*projects*/{*id*}/*issues*/
Commentaires	/*projects*/{*id*}/*issues*/{*id*}/*comments*/

Actions

Création	POST
Accès	GET
Mise à jour	PUT
Suppression	DELETE

Démonstration avec POSTMAN

Démonstration

- Démonstration de la documentation.
- Démonstration de l'API.

Documentation

Pourquoi documenter ?

Plusieurs raisons de maintenir la documentation d'un projet

- Le projet peut être co-développé par de nombreux contributeurs.
- Centralise les informations importantes.
- Les contributeurs du projet peuvent être amenés à changer :
- Facilite la formation des nouveaux développeurs.
- Cela peut permettre une traçabilité des fonctionnalités.

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.

Où et comment documenter l'API ?

- Utilisation de sphinx
- Utilisation de Postman
- Autre possibilité : *github* propose des pages web par projets ainsi qu'un wiki.

Documenter le code

Sphinx

- Utilisé 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 [1] :

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

1. `https://owasp.org/www-community/Threat_Modeling`

OWASP

Top 10

- *Open Web Application Security Project*.
- Propose (entre autres) un classement des 10 risques de sécurités les plus critiques.

OWASP 2021

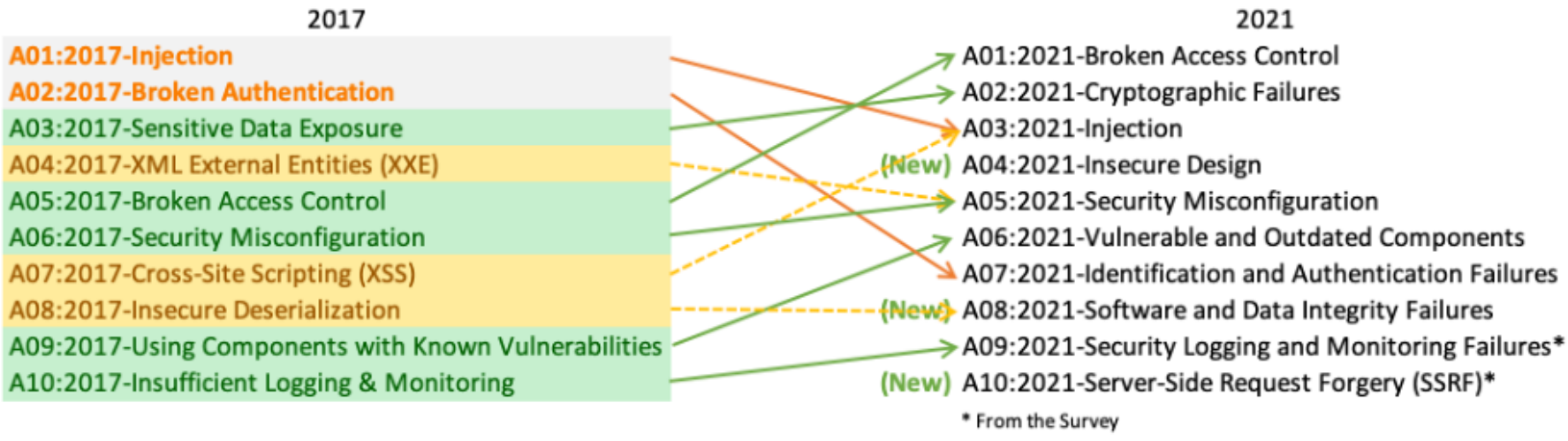
Le top 10 des menaces de sécurité

Version 2021 du Top 10 OWASP [2] :

1. Contrôles d'accès défaillants.
2. Défaillances cryptographiques.
3. Injections.
4. Conception non-sécurisée.
5. Mauvaise configuration de sécurité.
6. Composants vulnérables et obsolètes.
7. Identification et authentification de mauvaise qualité.
8. Manque d'intégrité des données et du logiciel.
9. Carence des systèmes de contrôle et de journalisation.
10. Falisification des requêtes côté serveur.

2. https://owasp.org/Top10/fr/

Comparaison avec l'OWASP 2017



Sécurité mise en place par Django

Protection contre les attaques XSS

Le système de *template* de django échappe certains caractères **mais** ne protège pas contre toutes les attaques XSS [3].

- *via* du code HTML et JS dans les variables
- *via* du code HTML en base de données

Protection contre les attaques CSRF

Protection par le *middleware* CSRFViewMiddleware. Cependant il faut être prudent lors de l'utilisation du décorateur csrf_exempt.

Protection contre les injections SQL

À moins que l'on n'écrive des requêtes en SQL brutes, le système de paramétrisation apporte une protection aux injections SQL.

3. `https://docs.djangoproject.com/fr/4.1/topics/security/`

Sécurité et configuration de Django

Autres protections [4]

- Détournement de clic (*via* les *frames* HTML).
- SSL et HTTPS.
- Validation de l'entête HTTP "Host".
- Sécurité des sessions.
- ...

Mise en production

Django propose une commande permettant de vérifier la sécurité d'une application web avant sa mise en production.
./manage.py check –deploy

4. https://docs.djangoproject.com/fr/4.1/topics/security/

Renforcer la sécurité

Sécurités supplémentaires

- Authentification par nom d'utilisateur et mot de passe :
  - Vérification de la sécurité du mot de passe.
- 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 Django

Il est possible d'utiliser les tests Django pour vérifier la sécurité du système développé. Il existe différents niveaux de tests : ici nous parlons de tests non fonctionnels d'acceptations et non pas de tests fonctionnels unitaires ($\rightarrow$ en complément, non pas à la place).

Respect de la RGPD

Quelques points importants pour le respect de la RGPD

- Les données personnelles sont chiffrées dans la base de données.
- Un utilisateur connecté peut actualiser ou supprimer toutes les ressources dont il est l'auteur.
- Les données personnelles stockées en base de données sont réduites.
- Après publication, il doit être possible pour un utilisateur de demander la suppression de toutes ses données personnelles.
- $\rightarrow$ C'est une question de droit, la consultation d'un juriste est souvent conseillée.

$\rightarrow$ Voir `https://www.economie.gouv.fr/entreprises/reglement-general-sur-protection-des-donnees-rgpd`

Développement

Vue d'ensemble

Applications

- authentication : pour gérer les utilisateurs et l'authentification.
- project : pour gérer les projets et ce qui tournent autour.

L'authentification

- Authentification par jeton JWT.
- Utilisation de la bibliothèque djangorestframework-simplejwt.
- Fournit une vue toute faite TokenObtainPairView.

```
...
from rest_framework_simplejwt.views import TokenObtainPairView

urlpatterns = [
    ...
    path('login/', TokenObtainPairView.as_view(), name='login')
]
```

Les rôles

Plusieurs rôles possibles

- Collaborateur d'un projet :
  - Contributeur (*contributor*)
  - Responsable (*supervisor*)
  - Auteur (*author*)
- Un utilisateur peut avoir plusieurs rôles !

Modèle d'un collaborateur

```
class Collaborator(models.Model):
    SUPERVISOR_ROLE = 'SUPERVISOR'
    CONTRIBUTOR_ROLE = 'CONTRIBUTOR'
    AUTHOR_ROLE = 'AUTHOR'

    ROLES = [
        (SUPERVISOR_ROLE, 'supervisor'),
        (CONTRIBUTOR_ROLE, 'contributor'),
        (AUTHOR_ROLE, 'author')
    ]

    user = models.ForeignKey(User, on_delete=models.CASCADE)
    project = models.ForeignKey(Project, on_delete=models.CASCADE)
    role = models.CharField(max_length=256, choices=ROLES)
```

Les projets

Un projet c'est :

- Un titre.
- Une description.
- Un type : ANDROID, IOS, FRONTEND ou BACKEND.

```
class Project(models.Model):
    ANDROID_TYPE = 'ANDROID'
    IOS_TYPE = 'IOS'
    BACKEND_TYPE = 'BACKEND'
    FRONTEND_TYPE = 'FRONTEND'

    TYPES = [
        (ANDROID_TYPE, 'Android'),
        (IOS_TYPE, 'IOS'),
        (BACKEND_TYPE, 'Back-end'),
        (FRONTEND_TYPE, 'Front-End')
    ]

    title = models.CharField(max_length=256)
    description = models.CharField(max_length=4096)
    type = models.CharField(max_length=64, choices=TYPES)
```

Modèle d'un problème

```
class Issue(models.Model):
  # ...

  title = models.CharField(max_length=256)
  description = models.CharField(max_length=4096)
  tag = models.CharField(max_length=128, choices=TAGS)
  priority = models.IntegerField()
  status = models.CharField(max_length=128, choices=STATUS)
  project = models.ForeignKey(Project, on_delete=models.CASCADE)
  author = models.ForeignKey(
      User,
      on_delete=models.CASCADE,
      related_name='author'
  )
  assignee = models.ForeignKey(
      User,
      on_delete=models.CASCADE,
      related_name='assignee'
  )
  created = models.DateTimeField(auto_now=True)
```

Tags et status

```
class Issue(models.Model):
    BUG_TAG = 'BUG'
    IMPROVEMENT_TAG = 'IMPROVEMENT'
    TASK_TAG = 'TASK'

    TAGS = [
        (BUG_TAG, 'bug'),
        (IMPROVEMENT_TAG, 'improvement'),
        (TASK_TAG, 'task')
    ]

    OPEN_STATUS = 'OPEN'
    CLOSED_STATUS = 'CLOSED'

    STATUS = [
        (OPEN_STATUS, 'open'),
        (CLOSED_STATUS, 'closed')
    ]

    # ...
```

Modèle d'un commentaire

```
class Comment(models.Model):
    description = models.CharField(max_length=4096)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    issue = models.ForeignKey(Issue, on_delete=models.CASCADE)
    created = models.DateTimeField(auto_now=True)
```

Sérialisation des modèles

Motivation

- Permet de passer d'une classe python au format JSON.
- Utilisation de la classe ModelSerializer de DRF.

```
class ProjectSerializer(ModelSerializer):
  class Meta:
      model = models.Project
      fields = [
          'id',
          'title',
          'description',
          'type'
      ]
```

Les vues

Les vues DRF

- Représentent une série de *endpoints* de l'API.
- Il y a une vue par type de ressource (projets, problèmes, commentaires etc).

Exemple : la vue des commentaires

Définition

```
class CommentView(mixins.CreateModelMixin,
                  mixins.UpdateModelMixin,
                  mixins.DestroyModelMixin,
                  mixins.ListModelMixin,
                  mixins.RetrieveModelMixin,
                  viewsets.GenericViewSet):
  serializer_class = serializers.CommentSerializer
  queryset = models.Comment.objects.all()
```

Exemple : la vue des commentaires

Permissions

```
class CommentView(mixins.CreateModelMixin,
                  mixins.UpdateModelMixin,
                  mixins.DestroyModelMixin,
                  mixins.ListModelMixin,
                  mixins.RetrieveModelMixin,
                  viewsets.GenericViewSet):

  # ...

  def get_permissions(self):
    if self.action in ['update', 'destroy']:
        return [
            rest_permissions.IsAuthenticated(),
            permissions.IsProjectRelated(),
            permissions.IsCommentAuthor()
        ]

    return [
        rest_permissions.IsAuthenticated(),
        permissions.IsProjectRelated()
    ]
```

Les types de permissions

Types

- IsProjectAuthor
- IsProjectRelated
- IsIssueAuthor
- IsCommentAuthor

Exemple

```
class IsProjectAuthor(BasePermission):
  def has_permission(self, request, view):
    user = request.user
    project = get_project(view)

    role = models.Collaborator.AUTHOR_ROLE
    return models.Collaborator.objects.filter(user=user,
                                              project=project,
                                              role=role).count() > 0
```

URL et routage

Django Rest Framework

- Permet de définir des routeurs permettant de servir une série d'URL directement à partir d'une vue DRF.
- **Ne permet pas** de gérer facilement les ressources imbriquées.
- La documentation de DRF [5] évoque une bibliothèque supplémentaire permettant justement de gérer ce cas de figure.

5. https://www.django-rest-framework.org/api-guide/routers/#drf-nested-routers

URL et routage

Django Rest Framework Nested Routers

- Permet de définir ressources imbriqués *via* un routeur NestedSImpleRouter.

```
projects = routers.SimpleRouter()
projects.register('projects', views.ProjectView, basename='projects')

users = routers.NestedSimpleRouter(projects, 'projects',
                                   lookup='project')
users.register('users',
               views.UserView,
               basename='users')

issues = routers.NestedSimpleRouter(projects, 'projects', lookup='project')
issues.register('issues', views.IssueView, basename='issues')

comments = routers.NestedSimpleRouter(issues, 'issues', lookup='issue')
comments.register('comments', views.CommentView, basename='comments')
```

Définitions des routes

Les routes

```
urlpatterns = [
  path('', include(projects.urls)),
  path('', include(users.urls)),
  path('', include(issues.urls)),
  path('', include(comments.urls)),
]
```

Conclusions

Travail effectué

- Mise en place d'une API REST testée avec des ressources imbriquées et un système d'authentification et de permissions personnalisées.
- Mise en oeuvre de principes de sécurités *via* Django.
- Publication d'une documentation *readthedocs* et Postman.

Pistes d'améliorations

- Déploiement de l'API sur un serveur.
- Adapter la stratégie de test en incluant certains risques du top 10 OWASP.
- Documenter les conventions de nommages et les choix techniques.

Merci pour votre attention