|  |
| --- |
| TPI  **Gestion du stock de la section Informatique** |

Alessandro D’Angelo

CIN4B ETML

Chef de projet : Dimitrios Lymberis

Experts : Bernard Oberson, Benjamin Wolf

Lieu : CFPV, Avenue de Valmont 28b, 1010 Lausanne

Date : du 05.05.2023 au 09.06.2023

Durée : 89 heures

**Table des matières**

[1 Analyse préliminaire 3](#_Toc136975572)

[1.1 Introduction 3](#_Toc136975573)

[1.2 Objectifs 4](#_Toc136975574)

[1.3 Planification initiale 4](#_Toc136975575)

[1.4 Méthode de projet 4](#_Toc136975576)

[2 Analyse / Conception 5](#_Toc136975577)

[2.1 Concept 5](#_Toc136975578)

[*2.1.1* Maquette 5](#_Toc136975579)

[*2.1.2* Base de données 7](#_Toc136975580)

[*2.1.3* Modèle - Vue - Template 9](#_Toc136975581)

[2.2 Stratégie de test 9](#_Toc136975582)

[3 Réalisation 10](#_Toc136975583)

[3.1 Dossier de réalisation 10](#_Toc136975584)

[3.1.1 Création du projet 10](#_Toc136975585)

[3.1.2 Architecture du projet 12](#_Toc136975586)

[3.1.3 Detail des modèles 14](#_Toc136975587)

[3.1.4 Choix / implémentation des URLs 14](#_Toc136975588)

[3.1.5 Les formulaires en Django 15](#_Toc136975589)

[3.1.6 Système d’authentification 17](#_Toc136975590)

[3.2 Description des tests effectués 18](#_Toc136975591)

[3.3 Erreurs restantes 18](#_Toc136975592)

[3.4 Liste des documents fournis 18](#_Toc136975593)

[4 Conclusions 19](#_Toc136975594)

[5 Annexes 20](#_Toc136975595)

[5.1 Résumé du rapport du TPI / version succincte de la documentation 20](#_Toc136975596)

[5.2 Sources – Bibliographie 20](#_Toc136975597)

[5.3 Journal de travail 20](#_Toc136975598)

[20](#_Toc136975599)

[5.4 Manuel d'Installation 20](#_Toc136975600)

[5.5 Manuel d'Utilisation 20](#_Toc136975601)

[5.6 Archives du projet 20](#_Toc136975602)

# Analyse préliminaire

## Introduction

Au sein de la filière informatique de l’ETML, il y a un grand nombre de matériaux différents utilisés dans le cadre de la formation des apprentis. Il peut s’agir d’outils portables comme des tablettes, d’outils réseau comme un routeur ou encore de périphériques tels que des écrans. Ces divers matériaux sont labelisés et stockés dans les différents laboratoires de l’établissement et peuvent être rangés dans des armoires.

Le recensement de ces divers équipements est aujourd’hui effectué à l’aide d’un tableau Excel auquel les données y sont rentrées manuellement.

Le but de ce projet est donc de réaliser une application web qui permet une gestion de l’inventaire et des emprunts facilitée.

La technologie choisie pour mettre en place l’application est la combinaison du langage Python avec le framework Django, qui permet la réalisation d’application web.

*Ce chapitre décrit brièvement le projet, le cadre dans lequel il est réalisé, les raisons de ce choix et ce qu’il peut apporter à l’élève ou à l’école. Il n’est pas nécessaire de rentrer dans les détails (ceux-ci seront abordés plus loin) mais cela doit être aussi clair et complet que possible (idées de solutions). Ce chapitre contient également l’inventaire et la description des travaux qui auraient déjà été effectués pour ce projet.*

Ce projet se déroule dans le cadre d’un Travail Pratique Individuel (TPI) et fait suite au projet d’approfondissement qui utilisait les mêmes technologies.

L’environnement de développement utilisé est le suivant :

* Un MacBook Air 2020 sous MacOS 13.2.1 Ventura
* Visual Studio Code
* La suite Microsoft Office 365
* Un s\_erveur Python
* Un dépôt GitHub

Les modules ayant été utile à la réalisation de ce projet :

* [302](https://www.modulbaukasten.ch/module/302/3/fr-FR?title=Utiliser-les-fonctions-avanc%C3%A9es-d%E2%80%99Office) : Utilisation avancée d’office
* [104](https://www.modulbaukasten.ch/module/104/3/fr-FR?title=Impl%C3%A9menter-un-mod%C3%A8le-de-donn%C3%A9es), 105, 151 : Schématiser, implémenter et gérer une base de données
* [306](https://www.modulbaukasten.ch/module/306/4/fr-FR?title=R%C3%A9aliser-de-petits-projets-dans-son-propre-environnement-professionnel) & [426](https://www.modulbaukasten.ch/module/426/1/fr-FR?title=D%C3%A9velopper-un-logiciel-avec-des-m%C3%A9thodes-agiles) [431](https://www.modulbaukasten.ch/module/431/2/fr-FR?title=Ex%C3%A9cuter-des-mandats-de-mani%C3%A8re-autonome-dans-son-propre-environnement-professionnel): Gestion de projet
* 120, 318, & [411](https://www.modulbaukasten.ch/module/411/1/fr-FR?title=D%C3%A9velopper-et-appliquer-des-structures-de-donn%C3%A9es-et-algorithmes) : Développer et appliquer des structures de données et algorithmes

## Objectifs

Selon le cahier des charges les points suivants doivent être présents :

* L’interface s’adapte en fonction de la connexion (menu personnalisé, profil utilisateurs)
* Il est possible d’ajouter, modifier, supprimer un article et visualiser le détail de celui-ci.
* Il est possible d’ajouter, modifier, supprimer un emprunt et visualiser le détail de celui-ci
* Il est possible d’afficher une liste d’articles avec filtre et vision si l’article est emprunté
* Affichage sous forme de liste de tous les articles empruntés actifs et ceux de l’utilisateur connecté. Les deux listes triées selon la date d’emprunt plus récente.
* Génération d’un QR code pour un article et impression selon un format réglable
* La modélisation de la base de données qui respecte la nomenclature Merise et le MCD / MLD / MPD sont présents et corrects

## Planification initiale

La planification initiale est fournie en [annexe](#_Annexes).

## Méthode de projet

La méthode des 6 pas a été utilisée, l’agilité n’étant pas nécessaire dans ce projet. Elle consiste en 6 points définis comme tels :

1. **S’informer**

La première étape se focalise sur la compréhension des tâches, définir les objectifs et besoins du projet.

1. **Planifier**

Phase de planification initiale et détaillée.

1. **Décider**

Phase de prise de décision selon le cahier des charges.

1. **Réaliser**

Étape de réalisation des tâches définies.

1. **Contrôler**

Tester le bon fonctionnement de l’application et de son implémentation selon les directives du cahier des charges.

1. **Évaluer**

Passer en revue ce qui a été réalisé avec le chef de projet et les experts.

# Analyse / Conception

## Concept

### Maquette

La maquette permet d’avoir une vue indicative de la disposition des différents éléments dont ont besoin chaque page afin d’anticiper la structure du code.

#### Squelette du site

Le menu, le bandeau ainsi que le pied de page restent les mêmes à travers les pages, seule la partie de contenu change.

*Bandeau*

*Bloc menu*

*Pied-de-page*

*Contenu*

#### Authentification & état connexion

A screenshot of a login form

Description automatically generated with low confidenceEn haut à gauche de chaque page, dans le bandeau, est présent un état de connexion. Si un utilisateur est connecté, il sera affiché son nom d’utilisateur, son nombre d’emprunts en cours ainsi que d’un bouton de déconnexion. Si personne n’est connecté, seul un bouton de connexion s’affiche.



#### A screenshot of a computer Description automatically generated with medium confidencePage d’accueil

#### A screenshot of a computer Description automatically generated with low confidencePage de produit

#### Formulaire d’ajout / ?

### Base de données

*A picture containing text, screenshot, diagram, font

Description automatically generated*Pour la base de données, Le MCD à été schématisé à l’aide de DB-Main afin d’avoir une idée visuelle et réfléchie des differentes entités et des possibles relations entre elles :

t\_products :

Elle contient un article qui fait partie d’une catégorie faisant elle-même partie d’un type . Un produit est doté d’un nom, d’une image, d’une note/description ainsi que d’un prix et d’une date d’achat.

t\_types :

Contient le nom d’un type (exemple : Réseau).

t\_categories :

Contient le nom d’une catégorie (exemple : Routeur).

t\_article :

Un article est un produit ayant été labelisé (ici stocké dans « artName ») et qui peut être en premier lieu situé dans un labo et a la possibilité d’être dans une armoire de ce dernier. Un article peut être emprunté par un utilisateur, auquel cas les informations suivantes seront enregistrées :

* Un emplacement de location (Exemple : maison ou office)
* La date de l’emprunt
* La date de retour de l’emprunt
* Une note liée à l’emprunt (exemple : État de l’article au moment de l’emprunt)

t\_user :

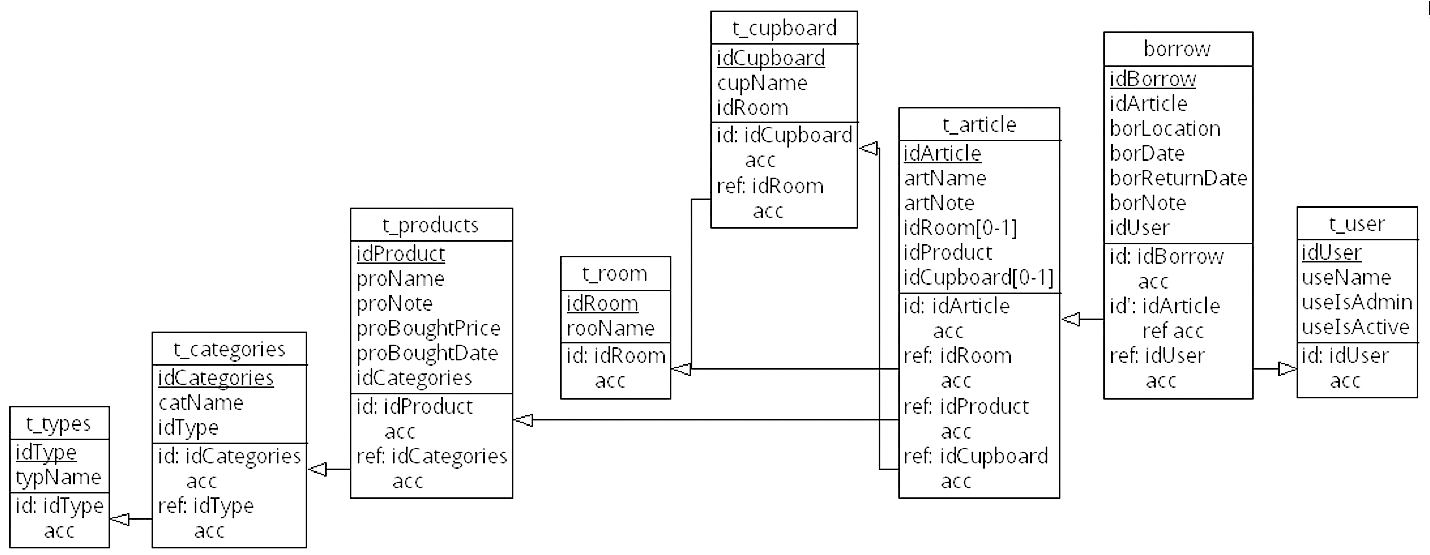
Ici sera stocké le nom de l’utilisateur, s’il est administrateur et s’il est toujours actif.

t\_room :

Contient le nom d’une salle/labo (exemple : A12) qui peut être ou ne pas être doté d’une ou plusieurs armoires.

t\_cupboard :

Le nom d’une armoire (exemple : ARM-201) qui est obligatoirement située dans une salle.

Le schéma MCD a ensuite été converti en MLD afin de visualiser les relations entre les tables :

### Modèle - Vue - Template

Le MVT est une architecture propre à Django. Elle est similaire au MVC (Modèle - Vue - Contrôleur) qui est plus commun. La Vue du MVT diffère, elle agit plutôt comme un contrôleur, qui pourra ensuite remplir le fichier Template, qui serait donc l’équivalent de la vue du MVC. Voici un schéma réalisé détaillant le pattern MVT :   
//TODO : Schéma MVC,

// Ajouter description des schémas

Graphical user interface

Description automatically generated

## Stratégie de test

*Décrire la stratégie globale de test:*

Il est important de procéder à des tests afin de s’assurer du bon fonctionnement et ne pas détériorer l’expérience utilisateur.  
Afin d’organiser les tests, ils seront divisés comme tels :

* Le nom de la page du site où le test sera effectué
* Description du test à effectuer
* Le résultat attendu
* Le résultat obtenu
* Temps requis pour corriger et/ou finaliser

Les tests seront réalisés manuellement au fur et à mesure du développement, avant de faire une vérification à la fin du projet.

# Réalisation

## Dossier de réalisation

### Création du projet

Pour instancier notre application Django, il suffit de quelques opérations.

Création de la structure du projet :

django-admin startproject [projectName]

Création de notre application en elle-même, ici appelée « app\_gestock »

python3 manage.py startapp [appName]

Afin d’avoir accès au panel administrateur, la commande suivante est utilisée pour nous créer un compte utilisateur aux droits élevés :

python3 manage.py createsuperuser

Création d’un dossier “templates” dans le dossier de l’application qui contient les pages html requises. Un fichier « base.html » y est ajouté, fonctionnant comme fichier de base, sont contenu sera remplacé par d’autres pages.

{% load static %}

{% comment %} Base HTML file used as a base for any HTML page {% endcomment %}

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8" />

<title>[title]</title>

<link rel="stylesheet" href="{% static 'css/style.css' %}" />

<link

rel="icon"

type="image/x-icon"

href="{% static 'resources/favicon.svg' %}"

/>

</head>

<body>

{% comment %} Block will be replaced with content from other pages{% endcomment %}

{% block content %}

{% endblock %}

</body>

</html>

Ici, la page d’index remplace le bloc « content » de base.html

{% comment %} Template of the main page, using the 'base.html' file {% endcomment %}

{% extends 'base.html' %}

{% comment %} Block that replace the same one from the base file {% endcomment %}

{% block content %}

{% endblock content %}

### Architecture du projet

**A screenshot of a computer

Description automatically generated with medium confidence**Les dossiers et fichiers sont organisés comme suit, selon l’architecture Django par défaut

Ici ‘app\_gestock’ est l’application créé dans le projet (plusieurs applications peuvent être crées dans un seul projet)

‘templates’ est un dossier créé manuellement mais qui est géré et reconnu par Django pour les fichiers HTML correspondant au visuel

‘tpi\_gestock’ correspond au projet qui contient notre application. Ici se trouvent les paramètres globaux.

‘db.sqlite3’ est la base de données SQLite intégrée à Django

**3.2 Paramétrage**

Les configurations du projet sont effectuées dans le fichier 'settings.py'. Après création de l'application et avant de commencer à coder, il est nécessaire de déclarer la création de l'application dans la constante qui gère les applications reconnues par Django.

Toutes ces applications sont par défaut pris en compte par Django lors de la création de notre projet.

La déclaration de l’intégration de TailwindCSS a été rajouté.

A screenshot of a computer program

Description automatically generated with low confidenceIl suffit maintenant d’entrer la commande suivante afin de pouvoir utiliser le framework tailwind :

pip install django-tailwind

Puis cette commande lors du développement de l’application afin de prendre en compte les derniers changements :

Python manage.py tailwind start

### Detail des modèles

Les Modèles en Django sont nécessaires au bon fonctionnement du projet car ceux-ci représentent la structure de la base de données et permet d’interagir avec cette dernière. Notez que la base de données est ici en SQLite, base de données générée et utilisée par défaut par Django.

Les Modèles sont situés dans un fichier nommé ‘models.py’, comme détaillé ci-dessus.

Voici un exemple de modèle :

Models.py

class t\_products(models.Model):

"""

Table representing a product.

"""

idProduct = models.AutoField(primary\_key=True) # Unique id for the product.

proName = models.CharField(max\_length=75, unique=True) # Name of the product.

proNote = models.CharField(max\_length=250) # Note for the product.

proImage = models.CharField(max\_length=500) # Image link of the product.

proBoughtPrice = models.FloatField() # Price at which the product was bought.

proBoughtDate = models.DateField() # Date on which the product was bought.

# Foreign keys to the corresponding tables

fkArticle = models.ForeignKey(t\_article, on\_delete=models.CASCADE, null=True)

fkCategory = models.ForeignKey(t\_categories, on\_delete=models.CASCADE)

fkType = models.ForeignKey('t\_types', on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return self.proName # Return the name instead of the object itself

### Choix / implémentation des URLs

Concernant les URLs, Django crée par défaut dans le projet un fichier nommée ‘urls.py’, ici il a été décidé pour rendre le code plus synthétique et plus distingué de mettre les URLs de l’application seulement dans celle-ci et pas dans le projet, c’est pourquoi les URLs du projet ressemble à cela :

*urls.py du projet*

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('app\_gestock.urls')), # Get urls from the app : app\_gestock

]

Seul le lien pour la configuration de l’interface admin de django est laissé ici car c’est celui du projet. L’autre lien inclus le second fichier d’URLs, celui créé dans l’application.

Dans l’application, ce fichier nommé également ‘urls.py’ remplis la fonction principale de celle-ci, à savoir rediriger un lien d’une requête, sur une vue de ‘views.py’ menant sur la page correspondante ([voir détail MVT](#_Modèle_Vue_Template)).

*urls.py de l’application*

from . import views

# Import the path & include function from the django.urls module

from django.urls import path, include

# Define a list of URL patterns for the web application

urlpatterns = [

# Map the URLs to it's view function and give it a name

path('', views.index, name='index'),

path('product', views.product, name='product'),

path('article', views.article, name='article'),

path('borrow', views.borrow, name='borrow'),

path('borrows', views.borrows, name='borrows'),

path('products', views.products, name='products'),

path('articles', views.articles, name='articles'),

path('addProduct', views.addProduct, name='addProduct'),

path('editProduct', views.editProduct, name='editProduct'),

path('deleteProduct', views.deleteProduct, name='deleteProduct'),

path('addArticle', views.addArticle, name='addArticle'),

path('editArticle', views.editArticle, name='editArticle'),

path('deleteArticle', views.deleteArticle, name='deleteArticle'),

path('addBorrow', views.addBorrow, name='addBorrow'),

path('editBorrow', views.editBorrow, name='editBorrow'),

path('jsondata', views.jsondata, name='jsondata'),

# Include the authentication URLs provided by Django and prefix them with /accounts/

path('accounts/', include('django.contrib.auth.urls')),

]

### Les formulaires en Django

Le Framework Django possède deux implémentations possibles de formulaires : forms.Form et forms.ModelForm. Ce sont deux classes qui permettent de définir et de valider des formulaires HTML.

Le premier intitulé « forms.Form» est une classe de formulaire de base plus classique qui ne communique pas directement avec le modèle et où il faut définir manuellement chaque champ et ses règles de validation.

D'autre part, forms.ModelForm est une sous-classe de forms.Form qui fournit une génération automatique de formulaire basée sur le modèle.

forms.Form et forms.ModelForm fournissent toutes deux des méthodes pour valider les données du formulaire et rendre le formulaire en HTML.

Ici, modelForm a été utilisé afin de garantir une simplification du code et de la communication avec le modèle.

Voici le formulaire d’ajout et de modification d’un article :

class articleForm(forms.ModelForm):

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

class Meta:

model = t\_article

fields = ['artName', 'artNote','fkRoom','fkProduct', 'fkCupboard']

labels = {

'artName': 'Label',

'fkProduct': 'Produit',

'artNote': 'Note',

'fkRoom': 'Labo',

'fkCupboard': 'Armoire'

}

Dans le formulaire sont définis les champs requis présents dans le modèle avec le label correspondant.

Formulaire d’emprunt :

class borrowForm(forms.ModelForm):

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.fields['borDate'].initial = datetime.date.today()

self.fields['borReturnDate'].initial = datetime.date.today() + datetime.timedelta(days=30)

class Meta:

model = t\_borrow

fields = ['borLocation', 'borDate', 'borReturnDate', 'borUser', 'fkArticle']

labels = {

'borLocation': 'Lieu demprunt :',

'borDate': 'Date demprunt :',

'borReturnDate': 'Date de retour :',

'borUser': 'Emprunté par :',

'fkArticle': 'Article :',

}

widgets = {

'borDate': DateInput(format=('%Y-%m-%d'),attrs={'type': 'date', 'class': 'dateForm'}),

'borReturnDate': DateInput(format=('%Y-%m-%d'),attrs={'type': 'date', 'class': 'dateForm'}),

}

Nous définissons tout d’abord une date initiale d’emprunt, la date actuelle. La date de retour, elle, est par défaut 30 jours après l’emprunt.

Les champs et leurs labels sont aussi présents. La partie widget permet, comme son nom l’indique, l’utilisation d’un widget de choix de date.

### Système d’authentification

Ici, seule la fonctionnalité de connexion est nécessaire. L’ajout d’utilisateur se fait dans le panel administrateur. Il serait cependant possible d’ajouter les fonctionnalités restantes, l’implémentation utilisant le système fourni par Django.

Dans le fichier ‘urls.py’, est ajouté le lien menant à la connexion :

Urls.py de l’application

  # Include the authentication URLs provided by Django and prefix them with /accounts/

path('accounts/', include('django.contrib.auth.urls')),

La connexion faisant appel à la méthode de connexion de Django, il suffit de rajouter les pages HTML de connexion et de déconnexion :

login.html

{% if form.errors %} {# Check if there are errors in the form #}

<p>Your username and password didn't match. Please try again.</p>

{% endif %}

<form method="post" action="{% url 'login' %}"> {# Login form #}

{% csrf\_token %}

<table>

<tr>

<td>{{ form.username.label\_tag }}</td>

<td>{{ form.username }}</td>

</tr>

<tr>

<td>{{ form.password.label\_tag }}</td>

<td>{{ form.password }}</td>

</tr>

</table>

<input type="submit" value="login">

<input type="hidden" name="next" value="{{ next }}">

</form>

logout.html

<p>Logged out!</p>

<a href="{% url 'login'%}">Click here to login again.</a>

## Description des tests effectués

*Pour chaque partie testée de votre projet, il faut décrire:*

* *les conditions exactes de chaque test*
* *les preuves de test (papier ou fichier)*
* *tests sans preuve: fournir au moins une description*

## Erreurs restantes

*S'il reste encore des erreurs:*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

## Liste des documents fournis

*Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions*

* *le rapport de projet*
* *le manuel d'Installation (en annexe)*
* *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
* *autres…*

# Conclusions

*Développez en tous cas les points suivants:*

* *Objectifs atteints / non-atteints*
* *Points positifs / négatifs*
* *Difficultés particulières*
* *Suites possibles pour le projet (évolutions & améliorations)*

# Annexes

## Résumé du rapport du TPI / version succincte de la documentation

## Sources – Bibliographie

*Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)… Et de toutes les aides externes (noms)*

## Journal de travail

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Durée** | **Activité** | **Remarques** |
|  |  |  |  |
|  |  |  |  |

## A picture containing text, screenshot, plot, diagram Description automatically generated

## Manuel d'Installation

## Manuel d'Utilisation

## Archives du projet

*Media, … dans une fourre en plastique*