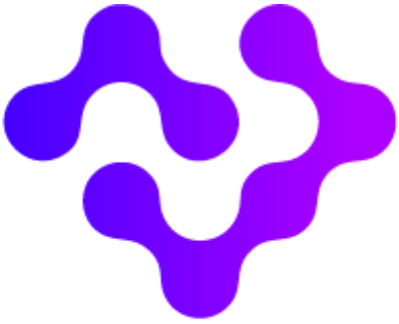
Internship report – Pallad

Hacker Ouvert – Cybersecurity learning platform



HACKER OUVERT

Théodore Billotte – Epitech Tek 3

March 31st – July 31st, 2025

Backend developer

Company – **Pallad**, Cap Oméga, Montpellier

Internship Supervisor – **Florian Lennuyeux**

Epitech Supervisor – **Arnaud Michel**

Table of contents

Introduction

This report aims to explain in detail everything I did during my four-month internship at Pallad, a cyber-security company in Montpellier. The focus of the internship was to develop their new project, Hacker Ouvert, a cybersecurity platform that aims to make cybersecurity accessible to both students and professionals.

In this report, I'm going to present the company and the project organization, structure, and history in the first section. In the second one, I'm going to explain in an email why the company should take me back for another internship.

The first section is going to be written as if it was an intern company documentation, and not as an internship report.

Section 1 – Onboarding Guide

Company Context

Florian Lennuyeux and **Cyril Garnier** were working at **Devensys**, a **cybersecurity** company based in **Montpellier**. They left the company after a few years working there, and they **founded Pallad**.

Pallad was **founded** very recently, in **July 2024** by **Cyril** and **Florian**. The head office is based at **Cap Oméga**, Montpellier. This company aims to help other companies with **email security**, **cyber diagnosis**, and **compliance assistance**. They are also doing training in schools like **EPSI** and **YNOV** in Montpellier.

The main subject of this document is going to be **Hacker Ouvert** as it is the only **Pallad's** technical project currently. This project was **originally** an **association** founded in **July 2017** that aimed to create a **discord community** to **share cyber information**, or **job opportunities**. Then, it **evolved**, and they started to make **formations for schools**, then they started to **intervene in events** to do some **cyber sensibilization**. Now, they **created** a real **Hacker Ouvert company** that keeps this idea of **community** and to **share the knowledge for free**, and to make **cybersecurity accessible** to both students and professionals by offering them diverse ways to learn with **lessons**, **practical work**, or **certifications** that could help them with a future job.

The Pallad team is composed of 5 people:

- **Cyril Garnier** – Co-founder and act as the administrative and financial director for Pallad, when working for companies, he can help with governance, risk and compliance and the risk analysis. He also took another role with Hacker Ouvert, he is doing the artistic direction and the frontend of the application.
- **Florian Lennuyeux** – Co-founder and acts as sales and training manager, he is also a teacher for Hacker Ouvert in schools and manages the system architecture for Hacker Ouvert.
- **Marie Guardiola** – Work-study marketing and communication, visuals, infographics, events preparation, and marketing strategy. She's also doing the Hacker Ouvert mockup with Cyril.
- **Yoan Benabdallah** – Sales, charged to generate leads for Pallad by reaching some companies and getting their contact in an excel file.
- **Théodore Billotte** – Backend Intern.

Onboarding and Working Process

Onboarding

Onboarding

When you arrive, you'll have to activate your **Microsoft account** created by your manager. Do the same for your **Scaleway account**; you should have received an email to activate your account. Otherwise, ask your manager to create an account. If your email is activated, you can create a **GitHub account** specially for Pallad. Once you do that, ask your manager to invite you to the **GitHub Organization**.

There is some information you're going to provide for the HR department to register you with the HR registry, and to properly get your salary. You need to provide your **banking details (RIB)** and your **social security number**.

Working Time

The **working hours are flexible**, but generally **start** around **9:30 AM**, and **end** around **5:30 PM**. You may have around **two hours for lunch** from **midday to 2 PM**.

These **schedules are very flexible**, so if you need to **start later some days**, it may be possible, you just need to **talk with your manager about it**. If someday you've **finished everything** you had to do, and you don't know what to do, you can just tell that to your manager, and **you can leave**. **Don't stay at work just to be physically present**; it's useless.

Remote

You can do **remote work as you want**, you just need to indicate it on your **outlook calendar**, but **remote is not holiday**, you must work as if you were at the office.

It's **better** if you come to the **office when everybody else is there**, and its way less util to come if you're alone at the office.

Lunch

If you don't want to spend money, you can **bring your own meal** and eat at the office, you can eat in **front of your computer**, or there are dedicated spaces at **Cap Oméga, with an outside terrace** for sunny days; and it happens a lot as we are at Montpellier.

As we are located at Cap Oméga, we're next to Dell's traffic circle, and there are a lot of **company restaurants**, but they are often **expensive for restaurant tickets**.

We tried some restaurants around the office, and I can give you some advice:

- **Boulangerie Ange**: Bakery with some cheap and expensive menus depending on your hunger, they also do some menu with a quarter of pizza instead of a sandwich, or an entire one if you are hungry; may vary between 6€ for cheap to more than 10€ for expensive.
- **Pâte Man**: Chinese specialties; around 12€ for a meal.
- **La Churrasqueira**: Chicken with two side dishes such as French fries, green beans, rice; around 12€ for a meal.
- **Burger University**: Good fast food; a bit expensive, around 15€ for a menu, but nice with friends or colleagues.
- **Pasta Cosi**: Italian food, such as pizza, pastas, …; around 10€ for a meal.

Team and Project Organization

There are two teams working on **Hacker Ouvert**. As a **web application**, there are **front-end** and **back-end** teams. You can eventually work in these **two teams if you're a full stack developer**.

As it is a **cybersecurity project**, it's important to ensure that there are **no vulnerabilities** on the platform.

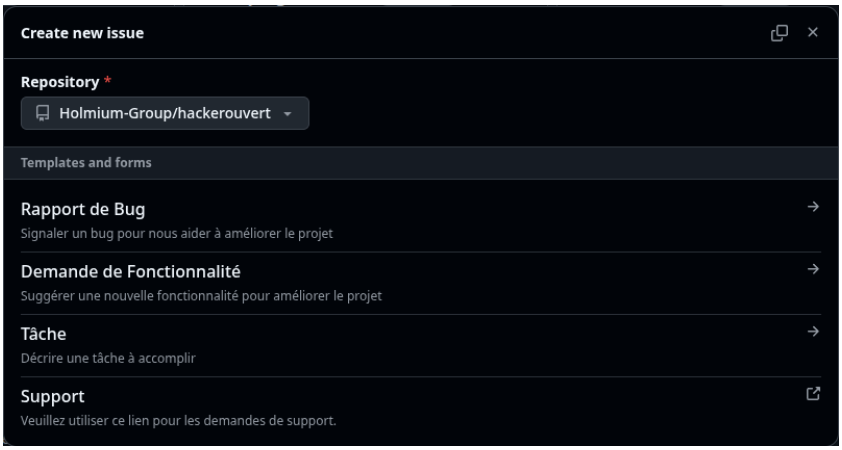
We are a **French organization**, we're trying to use only **made in France tools** or **open source**, this is the reason why we're using **Scaleway** for all our hosting, and we're trying to be **ecologic**.

Organization

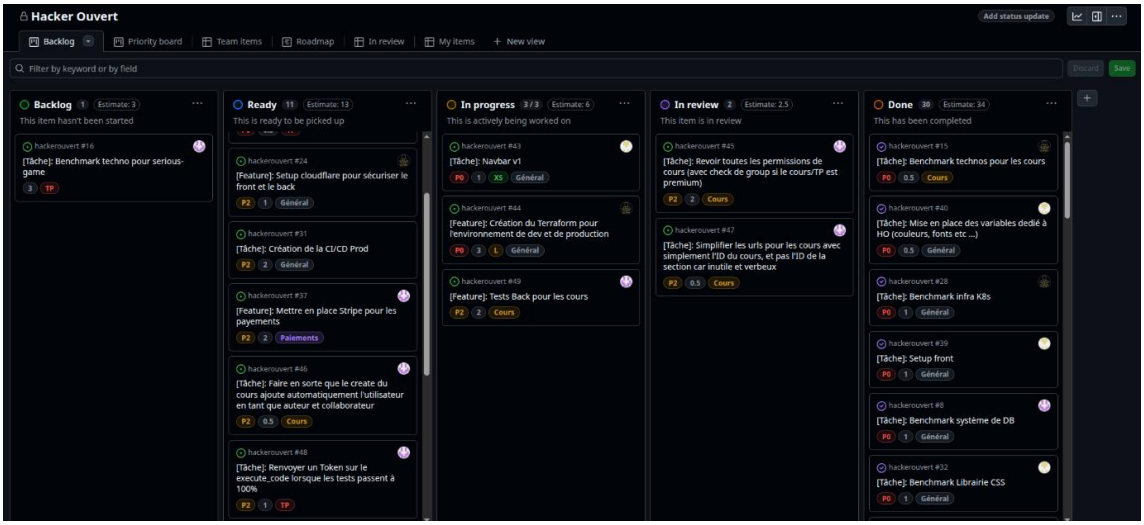
To organize the project development, we're using **GitHub Project** as a task management tool. We also use **Teams** for communication within the company.

The communications and the tickets are entirely in **French**, I don't really know why I'm doing this document in **English**, but it may have some educational purpose.

If you want to create a ticket on the **GitHub Project**, then you'll have to click the "**add item**" at the bottom of the column, type the name of your ticket and select the type of your issue by clicking on it. Now, you'll have to add a short description of your ticket, select the person who should realize the task, then click the "**create**" button. Once the ticket is created, you'll have to select the "**Epic**", the priority, and an estimated duration of the task (in days).



You'll have to communicate a lot to ensure what you're doing fits with the global vision. **Sharing ideas** is welcomed, and it can make the project even better with everybody's ideas. You can also **contest decisions** or **ideas** with arguments; you'll always be listened to if you do it well.



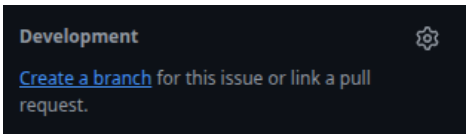
Collaboration

You're going to use GitHub as a version control system. On each project, you'll have two branches by default; **main** and **develop**, as indicated by their name, the main branch is the production one, and the develop is the development one.

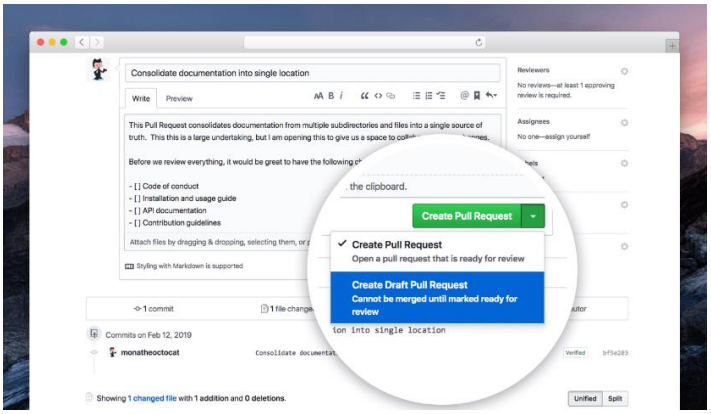
We have multiple GitHub projects inside the GitHub Organization for different purposes:

- **ho-front**: contains the front-end **code base** and a **Docker File** to build the project if needed.
- **ho-back**: contains the back-end **code base** and a **Docker File** to build the project if needed.
- **hackerouvert**: contains two submodules pointing to **ho-front** and **ho-back**; this is useful if you need to clone and run the app easily. We also have a **docker-compose** allowing us to start the app with one single command line.
- **ho-bot**: contains the source code of the different bots, we're currently have only a **discord bot** running on it. We also have a **docker-compose** to run all bots present in the repository.
- **tf-ho**: contains the **Terraform** configuration to deploy the infrastructure thanks to a **yaml** configuration.

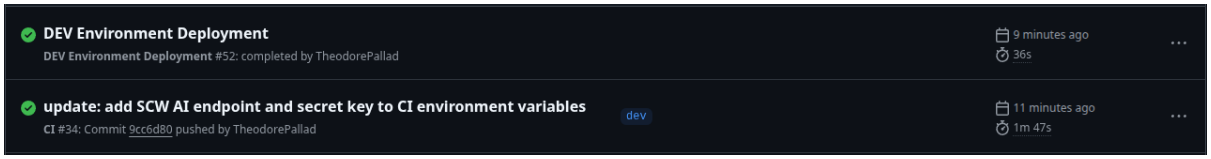
When you take a ticket on the GitHub Project, you can directly create the branch by clicking the "**Create a branch**" in the ticket.



It should create a branch named like this example: "49-feature-tests-back-pour-les-cours" and select the repository you want the branch to be created on. Once the branch has been created, pull the branch locally and start what you must do. When you finish your task, you can create a pull request to the **develop** branch. Another dev is going to check what you did, if there are no mistakes; If everything is ok, then you can merge your branch into development and **delete** your branch.



The **CI/CD** should have been triggered by your **push** to **develop**. You already should have run the tests locally, but if the **CI fails**, then you'll have to create a new ticket for a **hotfix** or return to your previous branch and make a pull request again.



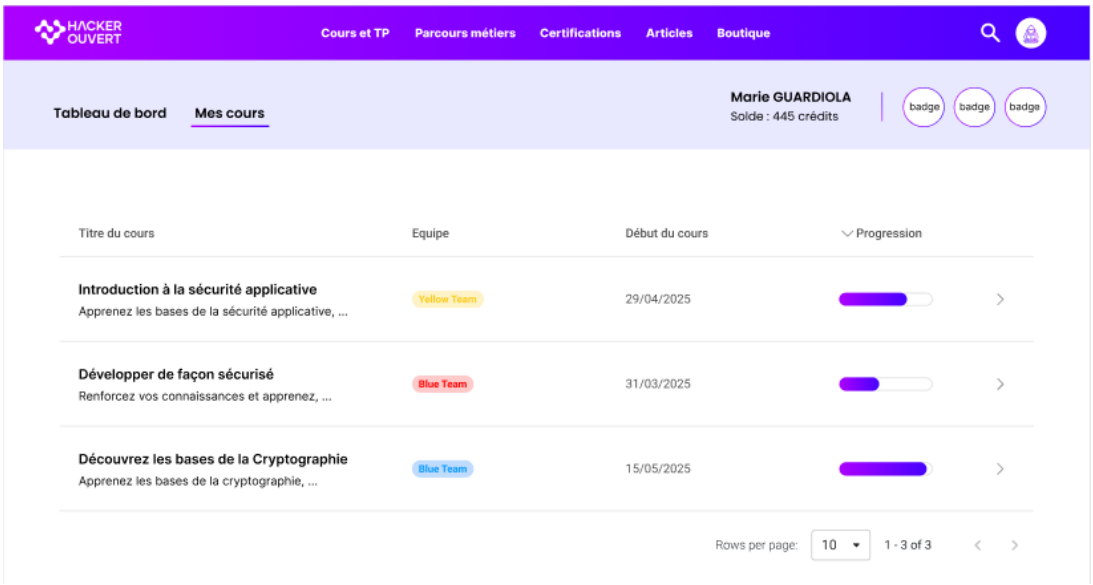
Otherwise, if the **CI** is passed, then the **CD** should have been triggered, and your code should be available in the **development** environment. You can test it manually, and you can see with your superior if you have one to push on main, or if you don't have one, you can do as you feel.

You may **write some documentation** for your **architecture** or just **write some explanations** about what you did on the **OneNote**, you can write everything that could be useful for actual or future colleagues, if they take your work back.

Front-end

Working on the Front-end team means creating mockups with **Figma** and integrating it into the application using **React** and **Tailwind CSS**.

You must discuss with the back-end team to know when there are changes to the **API** to be sure nothing has been broken in the front by those changes.



Back-end

The back-end mission is to develop the **API** for the front-end. This comes with **quality insurance**, by doing some **integration** and **functional tests**. This obviously comes with **GitHub actions** to test and deploy directly when you push something in the development or production branch.

Another mission of the backend is infrastructure. The entire infrastructure runs on **Scaleway** and is deployed thanks to **Terraform**. The development environment runs on a single machine with everything (front, back, database) directly integrated on the server to reduce costs. Otherwise, in the production environment, we're using a **Scaleway Postgres** database to store our app data. We're also using **S3 storage** for courses and practical works storage, as they are more voluminous files, it's better to store them in **S3** instead of SQL database.

Projects Overview and Setup

We're now going to see the **different repositories** you're going to work with and explain to you how to **run** and **deploy each project**.

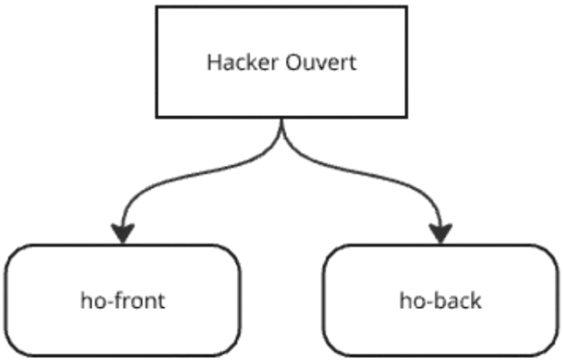
The commands shown only work for **Linux** and **Mac OS**. If you're using **Windows**, **you must find some alternatives** to these commands.

hackerouvert

About The Project

This is a meta project that contains **submodules** pointing to real project repositories, such as **ho-front** and **ho-back**.

The most important thing on this repository is the **docker-compose** file, because it's thanks to this that you'll be able to start the app easily.



Getting Started

Installation

Clone the repository and make sure that every submodule is up to date

```
1. git clone https://github.com/Holmium-Group/hackerouvert.git
2. cd hackerouvert git submodule update --init --recursive
```

Then, create a "**.env**" file to the root of the repository, and ask another developer or your manager for this information:

```
 1. DJANGO_SECRET_KEY=xxx
 2. DJANGO_DEBUG=True
 3. DJANGO_ALLOWED_HOSTS=0.0.0.0
 4. DJANGO_CSRF_TRUSTED_ORIGINS=0.0.0.0
 5. DJANGO_CORS_ALLOWED_ORIGINS=0.0.0.0
 6. SWAGGER_DEFAULT_API_URL=xxx
 7. POSTGRES_DBNAME=xxx
 8. POSTGRES_USER=xxx
 9. POSTGRES_PASSWORD=xxx
10. POSTGRES_HOST=xxx
11. POSTGRES_PORT=xxx
12. DISCORD_SECRET=xxx
13. DISCORD_BOT_TOKEN=xxx
14. DISCORD_API_URL=https://discord.com/api/v10
15. DISCORD_CLIENT_ID=xxx
16. DISCORD_CLIENT_SECRET=xxx
17. DISCORD_VERIFIED_ROLE_ID=xxx
18. DISCORD_GUILD_ID=xxx
19. REDIRECT_URI=xxx
20. SCW_ACCESS_KEY=xxx
21. SCW_SECRET_KEY=xxx
22. SCW_DEFAULT_ORGANIZATION_ID=xxx
23. SCW_DEFAULT_PROJECT_ID=xxx
24. SCW_DEFAULT_REGION=xxx
25. SCW_FUNCTION_TOKEN=xxx
26. SCW_FUNCTION_URL=xxx
```

Usage

Build and run the docker compose with this command

```
1. docker-compose up -d --build
```

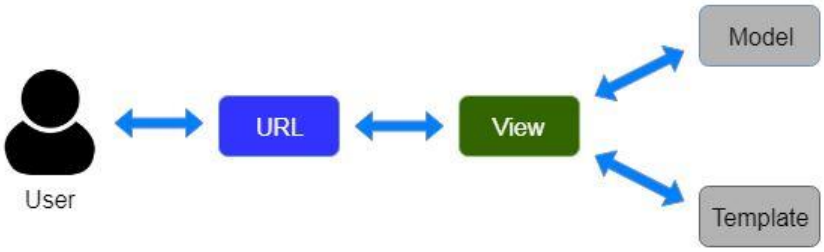
```
107  out: Successfully built d900806c7882
108  out: Successfully tagged hackerouvert_frontend:latest
109  err: hackerouvert_db_1 is up-to-date
110  err: Recreating hackerouvert_backend_1 ...
111  err: Recreating hackerouvert_backend_1 ... done
112  err: hackerouvert_frontend_1 is up-to-date
113  ============================================
114  ✅ Successfully executed commands to all host.
115  ============================================
```

ho-back

About The Project

This project is the back-end of Hacker Ouvert. It is developed with the framework **Django** in **Python**.

This framework uses an **MTV** (Model-Template-View) structure, which is a variant of the **MVC** (Model-View-Controller).



Models are the representation of **database objects** with a **python class**, and this is Django who's going to generate every SQL request following the python class indications.

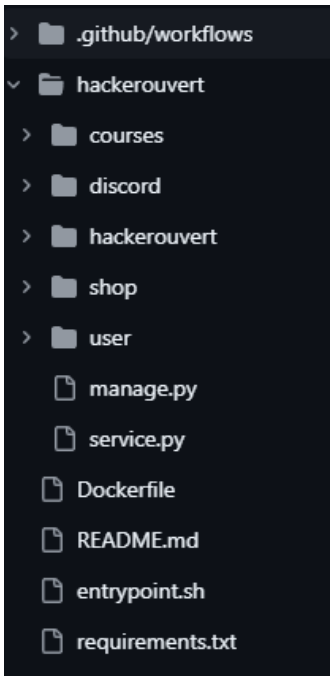
Templates are some **HTML reusable pages** with variables and logic, but as we're only working on the back end, we're not using them.

Views are the equivalent of the **Controllers** in the **MVC**, thanks to **Django** plugins like "djangorestframework" which make some abstractions and make the **CRUD Rest API** creation way easier with **Django**.

One of the greatest advantages of using Django is the admin panel that allows you to monitor and update every model of your application.

Structure

Django projects are structured with **project configuration** that contains **configs** and **URLs** to map applications to the project. And **applications**, where are the **models**, **views**, **serializers**, **admin panel**, and any **other file** you need for your API view.



Inside the **Hacker Ouvert project**, we already have some **applications**:

User: this app contains the model of the custom user, because Django have a user model per default, but it has been overwritten to add some custom properties like the achievements, the discord ID, or every premium course the user paid for. The views only allow the user to access and update his personal data, he can't access other users data.

Courses: this app is the core of Hacker Ouvert because it contains the entire logic and models for the courses, sections, lessons and practical works. Courses if the main object contains sections, and sections contain lessons and practical works. There are a lot of possible interactions and complex logic with courses, even the permission system is complex, because there is permission for the course creators, and only collaborators on a course can execute requests to update or remove the course. There are also some premium courses, and users must buy the course before accessing its content. This feature is primarily for practical work as it is for us the most expensive feature of our application.

Discord: this app contains no models, but interacts with the user's one, because it's used to link the discord account to the user and has some endpoints to check if a discord user is linked to discord, or just to give giveaway rewards to users (this is protected by a secret key to prevent any abuse).

Shop: contains models and a view to buying and get the purchase history of a user. There is also an article system to give the user access to some courses when bought.

Getting Started

Prerequisites

You must have **Python** installed on your computer.

Running this project **outside the Docker** should be useful if you need to debug the back end using your IDE properly without putting debug prints everywhere.

Installation

Clone and enter the repository

```
1. git clone https://github.com/Holmium-Group/ho-back.git
2. cd ho-back
```

Set up the python venv environment if you don't have one

```
1. python -m venv .venv
2. source .venv/bin/activate
```

Install the project requirements

```
1. pip install -r requirements.txt
```

Then, create a "**.env**" file to the root of the repository, and ask another developer or your manager for this information:

```
 1. DJANGO_SECRET_KEY="django-secret-key" # Change this to a random string
 2. DJANGO_DEBUG=True # True if you're in dev mode
 3. DJANGO_ALLOWED_HOSTS=0.0.0.0,127.0.0.1,127.0.0.0,localhost # Add your frontend host here
 4.  # PostgreSQL
 5. POSTGRES_DBNAME=hackerouvert
 6. POSTGRES_USER=user
 7. POSTGRES_PASSWORD=password
 8. POSTGRES_HOST=your-host
 9. POSTGRES_PORT=5432
```

Usage

Once you finish the installation, you can run the project by using this script, which is a script that applies migration, and run the server on port 8000

```
1. ./entrypoint.sh
```

You can also do it manually like that

```
1. python manage.py migrate
2. python manage.py runserver 0.0.0.0:8000
```

If you need to create a superuser account, just type

```
1. python manage.py createsuperuser
```

If you updated the models of your API, and you need to create a new migration, you can simply do that

```
1. python manage.py makemigration
2. python manage.py migrate
```

ho-front

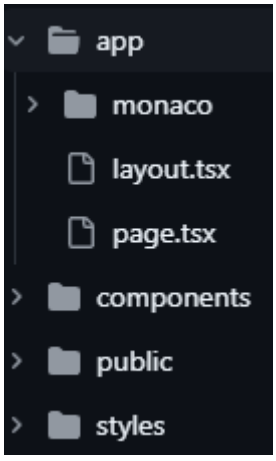
About The Project

This project is the front-end of Hacker Ouvert. It is developed with the frameworks **React** and **Tailwind CSS**.

You can access the **mockup** here.

Structure

The project structure is a classic **React** structure with **pages** represented by a folder and dedicated ".tsx" files. There are also components contained in the "**component**". Components are reusable snippets of code to use in other pages.



At the time I write this, the **front-end is in a very early version**, there is almost nothing on it, only the empty landing page, with the navbar, and the Monaco page that was used as a test for students. I don't even touch it a lot, because it's not my mission.

Getting Started

Prerequisites

You need the **back-end** to run if you want the front end to work properly. See here.

You need **NPM** to run this project.

Running this project outside a Docker may be useful if you want to debug your application, because you can run your code with a **debugger** directly in your IDE.

Installation

Clone the repository and enter it

```
1. git clone https://github.com/Holmium-Group/ho-front.git
2. cd ho-front
```

Install NPM packages

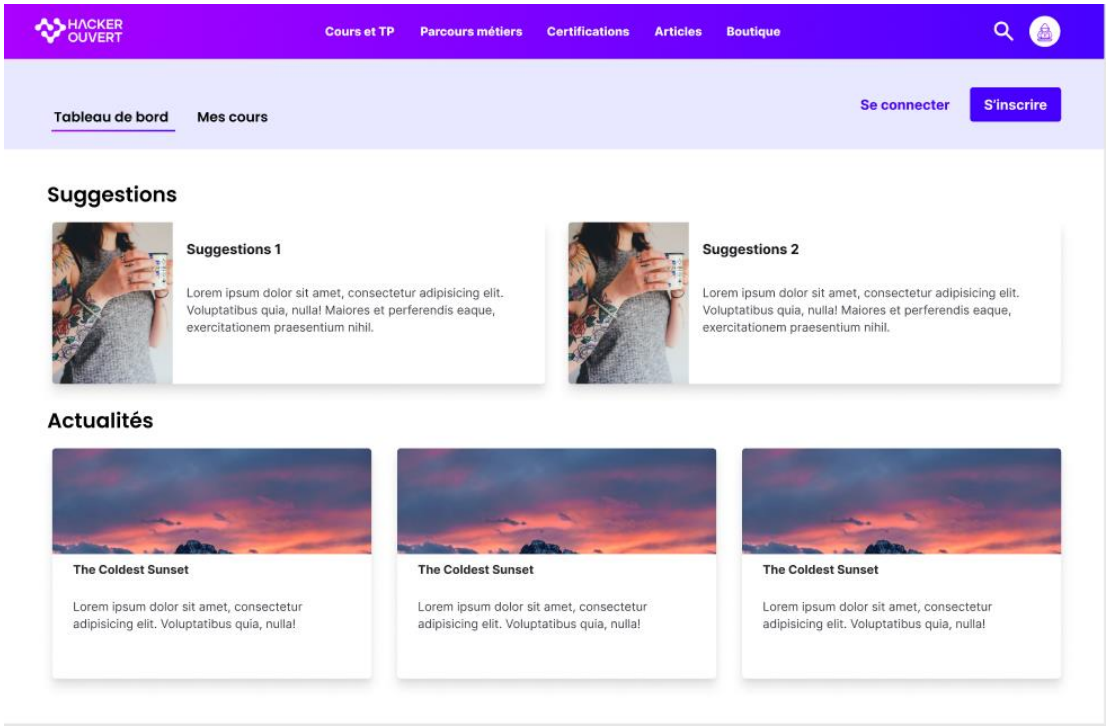
```
1. npm install
```

Then, create a "**.env**" file at the root of the repository, and ask another developer or your manager for required information. As the front-end has not really started yet, I can't give you any examples.

Usage

You can now run the project with this single line

```
1. npm run dev
```



ho-bot

About The Project

This project is supposed to contain every bot linked to **Hacker Ouvert**, but actually, there is **only one**, and it is a **discord bot**.

The discord bot has been developed in **python** using the **discord.py** library.

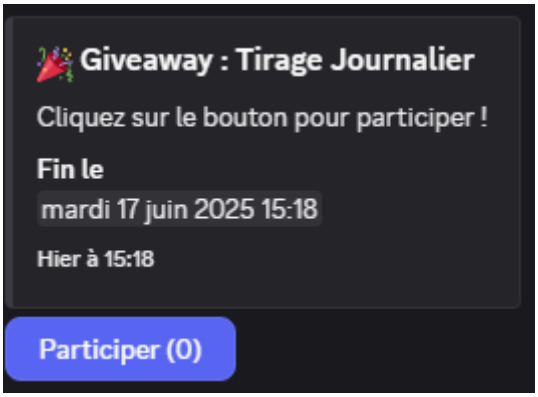
Structure

Discord.py has a system that can look like **Django**, but **instead of apps, it has cogs**, it's doing the same, but has a different name. **Cogs allow you to handle events, interactions, and create commands**.

We're also using the **SQL Alchemy python library as ORM** (Object-Relational Mapping) for the bot, because we need to store some data in a database, and it was easier that way.

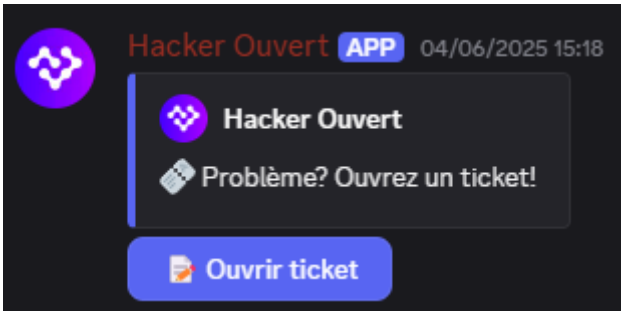
This bot is composed of 4 Cogs:

Giveaway: this is the most complex one, because you can configure giveaways using a Json configuration that allows you to schedule giveaways periodically after a certain period. You can also just start using a command and define manually what the reward should be. The system is directly linked to the backend; it means that only linked users can participate in giveaways. It also means that when the giveaway ends, it can directly give the reward to the user via the API.

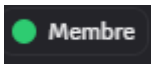


News: This is a simple system that fetches every 10 minutes RSS feeds and if new information's found, then it sends a message with a short description of the news and a link to the news with an image if the bot finds one in the feed.

Support: Allow linked customers to create a ticket and talk to the staff privately if the customer has any trouble or questions.



Verified: This is a system that's going to check if new members are already linked to our API, if it is, then the bot adds the role "Membre" to the user, and he can access the support and giveaways.



Getting Started

Prerequisites

The bot requires **python** to be installed.

Installation

Clone the repository and enter it

```
1. git clone https://github.com/Holmium-Group/ho-bot.git
2. cd ho-bot
```

Set up the python venv environment if you don't have one

```
1. python -m venv .venv
2. source .venv/bin/activate
```

Install python requirements

```
1. pip install -r requirements.txt
```

Then, create a "**.env**" file to the root of the repository, and ask another developer or your manager for this information:

```
1. DISCORD_TOKEN=xxx
2. DISCORD_GUILD_ID=xxx
3. DISCORD_SECRET_KEY=xxx
4. LOG_LEVEL=DEBUG
5. DATABASE_URL=postgresql://xxx:xxx@xxx:xxx/xxx
6. DB_HOST=xxx
```

```
DB_PORT=xxx
DB_NAME=xxx
DB_USER=xxx
DB_PASSWORD=xxx
MODERATOR_ROLE_IDS="[xxx]"
SUPPORT_CHANNEL_ID=xxx
TICKET_PURGE_DAYS=1
RSS_FEEDS=[""]
RSS_CHANNEL_ID=xxx
RSS_POLL_INTERVAL_MINUTES=10
TIMEZONE=Europe/Paris
API_BASE_URL=xxx
VERIFIED_ROLE_ID=xxx
```

Usage

You can now run the bot with this single line

```
python bot.py
```

Offboarding

Before leaving, you must be sure that **all your access has been revoked**, and your **accounts disabled or removed**. For that, you must spend some time with your manager to handle that situation properly.

You must have explained to your manager or colleagues every subject you've worked on, and it's even better if you could **write documentation** for it to explain to another developer how to pursue your work.

You shouldn't leave unfinished work, but if you have to, **you must talk about it to your manager**, because it may cause trouble, and block the team if you don't.

Here is the list of accounts to check:

- **Microsoft**
- **Scaleway**
- **GitHub**

Your team should have **prepared something** for your departure, to say **goodbye**, and to see your **colleagues one last time before leaving**.

Section 2 – Acceptance Letter

Hello, I hope you're doing well.

I'm sending you an email as I will soon need a **part-time job** or **internship** to complete my **5th year** at Epitech Montpellier. I am available from **September 7, 2026**, to **March 1, 2027**.

I saw that the launch of the 1st version of **Hacker Ouvert** was very successful, that's why I'd love to work on the next versions of **Hacker Ouvert**, a project I believe in, and has a good objective, because offering **courses for free**, or **cheap** price, is something great to me, and obviously, working with both of you, **Florian** and **Cyril** was a real **pleasure** to me, as you are **open** to the discussion, and **comprehensive**.

I think I can be very useful to you since I know exactly how the **backend** and **discord bot** work, and since I designed them myself. I also proved during my first internship that I am competent as a developer, and I can handle project organization properly.

During my first internship, I also helped with the **vision of the project**, I gave my own vision, and it helped you to make the project **evolve** for the **better**. I also **planned** and **organized** the entire development of the project, I created the **GitHub project**, and developed tools to help the organization, such as **issue templates** and the **CI/CD**.

Another point for taking me back is my **game-dev** experience with **Godot**, as you know, I released a game on Steam with Godot, and some of your practical works on the platform integrate serious games made with Godot, so it could be a great opportunity for you to enhance this part. I also have **cyber security skills** and knowledge that I could use to create some **content** on the platform, and thanks to my 4th year at Tsinghua, I now have some skills in **AI**, and I can help you to integrate that for the **future** version of the project.

I've attached my **CV**, if you need to remember me, or just as a **portfolio** for my **application**. Feel free to reach out directly via **email** or **discord**. I'd be more than happy to renew this experience at **Pallad**.

Have a great day,

Théodore Billotte

Conclusion

Working at **Pallad**, and more precisely, on the project **Hacker Ouvert** was a **great experience** for me. I **learned a lot of things** about **project management**, **infrastructure**, and even **web development**.

The internship is **not finished yet**, but I can say that I met a **little team** of 3 people (because I never saw Yoan), each of them has their **own skills**, and **tried to explore others instead** of just doing what they know, and I found that **great**, **everyone is very nice in the team**, and there is a **good work atmosphere**.

Since yet, I have done internships in 3 different companies, a **PME** (Mon Chasseur Immo), a **Minecraft Server** (Hyping) and a **Startup** (Pallad). I think that my next internship is going to be in a big company that seems great!

Thanks for reading this report. :)