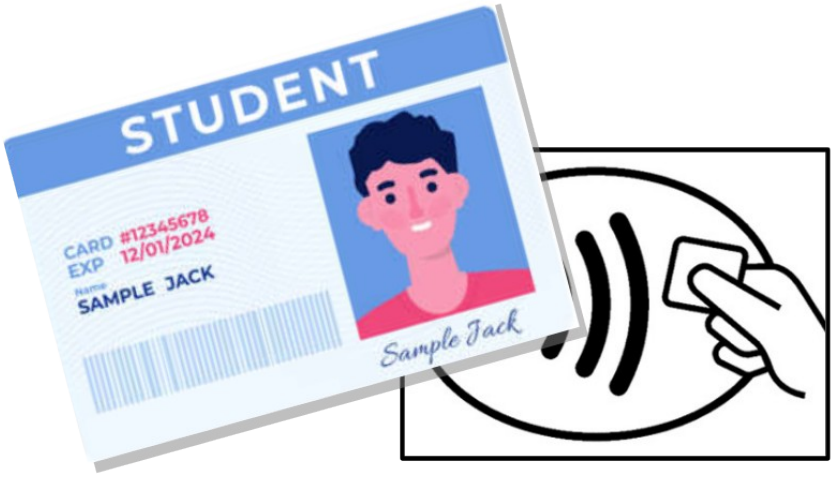
DÉPARTEMENT
Sciences Informatiques

Rapport technique du Projet BDD/Réseaux

Objectif : Réaliser le réseau, la base de données et le site d'un gestionnaire de présence pour une université



L3-Informatique Groupe A : Trinôme A8 (2024-2025)

Enseignant : M. Marc LEMAIRE

Adnane BOUYKNANE, Khalifa MEBARKI, Samy-Mohamed BOUAOUNI
28/11/2024 Version finale

Numéro de Versions de rapport	Date	Changements apportés
Version FInale	28 Novembre 2024	Ajout complète de la partie netcat, « Connexion BDD » et « site web ». Finalisation des parties « requêtes » et annexes contenant les codes. Révision complète du rapport .
Version 4	27 Octobre 2024	Ajout du “coeur” des codes réseaux (Client/Serveur) en annexe, ajout de 2 nouvelles tables + maj des schémas en conséquence. Modification du jeu de données.
Version 3	12 Octobre 2024	Modification du diagramme applicatif, ainsi que certaines positions d’éléments du rapport comme le sommaire, etc. Ajout du jeu de données.
Version 2	14 Septembre 2024	Finalisation du dictionnaire de données. Création du schéma E/A et relationnelle. Conversion des ébauches d’échanges réseaux en diagrammes applicatifs concrets.
Version 1	29 Septembre 2024	Création du Compte-Rendu et début du dictionnaire de données et échanges réseaux (ébauches).

1. Introduction

Actuellement en troisième année de licence Informatique à l'université CY Cergy Paris Université, nous allons réaliser un projet en base de données et réseau afin d'approfondir nos connaissances dans ces matières. Pour cela nous devrons nous documenter, faire des recherches internet, interroger nos professeurs et raisonner par nous-mêmes.

Pour compléter cette expérience, nous réaliserons donc un rapport de projet afin de vous décrire comment nous nous sommes organisés durant ce projet, mais également vous expliciter les problèmes auxquels nous avons dû faire face, ainsi que les solutions que nous avons pu trouver pour réaliser le sujet que nous avons choisi.

Dans un premier temps nous ferons la rédaction d'un rapport technique avec une courte description en quoi consiste le projet, le travail à effectuer, puis nous vous expliquerons les étapes du déroulement de ce projet.

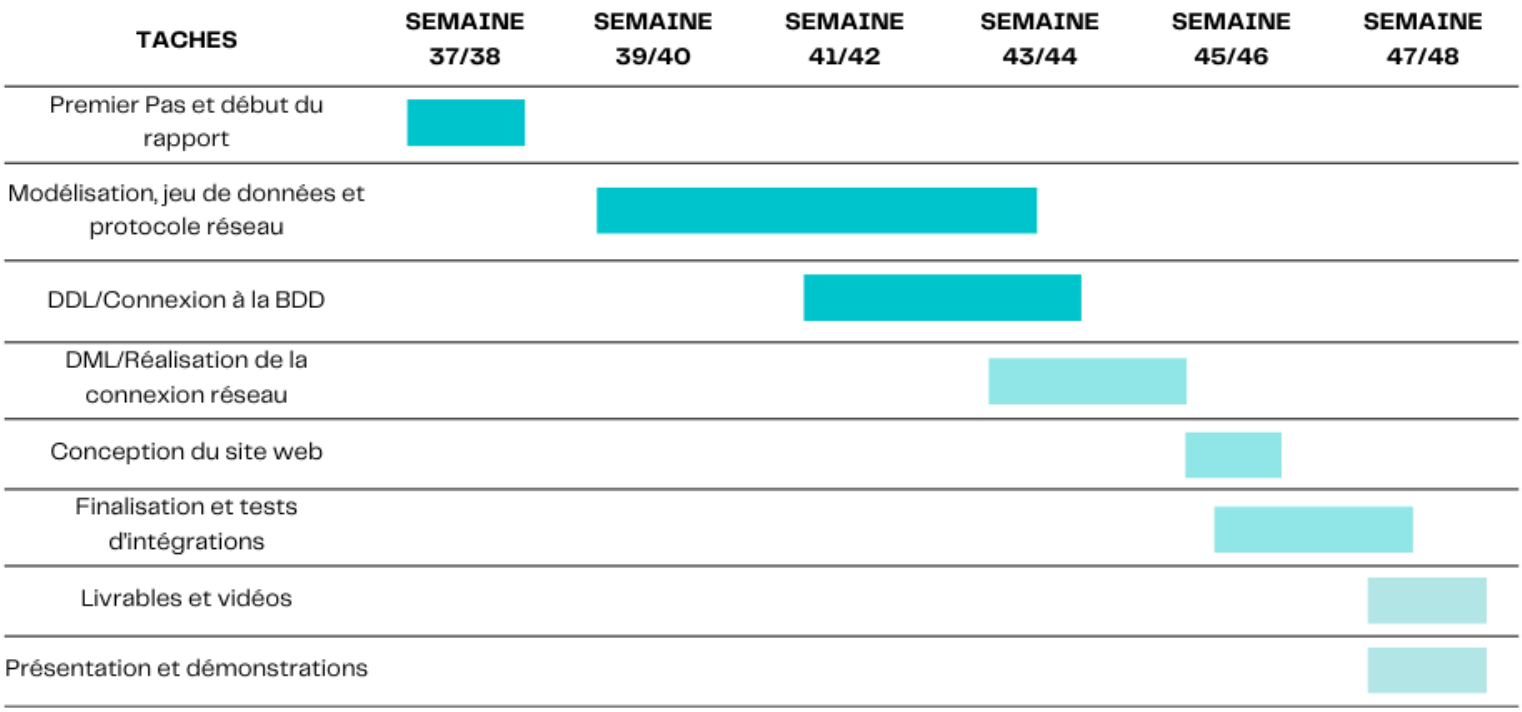
2. Contexte du projet

Description du projet

Pour ce projet, notre objectif est de créer une plateforme qui permet aux enseignants de consulter la présence des étudiants au sein de leurs cours. La gestion manuelle des présences est souvent lourde et sujette à des erreurs humaines. C'est là que notre application pour le suivi des présences des étudiants entre en jeu, visant à automatiser et faciliter ce processus.

La gestion des présences en classe constitue un défi quotidien pour les enseignants, nécessitant une attention constante. Notre projet se propose d'y répondre en créant une plateforme efficace permettant aux enseignants de vérifier la présence des étudiants rapidement et précisément. Les étudiants pourront enregistrer leur présence en passant leur carte étudiante sur un lecteur situé à l'entrée des salles de classe, ce qui réduira les tâches administratives et optimisera la gestion.

3. Répartition des tâches



4. Dictionnaire de données

Personnes	Type	Null/Non null	Exemple	Contraintes
id_personnes	CHAR(9)	NN	122001185, ...	Numérique, PK
nom	VARCHAR(60)	NN	MEBARKI	Alphabétique
prenom	VARCHAR(60)	NN	Khalifa	Alphabétique
num_telephone	CHAR(10)	NN	0768760148	Numérique, Unique
email	VARCHAR(64)	NN	Abc.de@xx.com	Unique, contient @, ...
adresse	VARCHAR(90)	NN	23 bd du port Cergy	Alphanumérique
date_naissance	DATE	NN	23/09/2001	> 1930 et < 2006
genre	ENUM	NN	Homme, Femme	
mot_de_passe	VARCHAR(70)	NN	mdp2024/E	

Etudiants	Type	Null/Non null	Exemple	Contraintes
id_persn_etudiant	CHAR(9)	NN	122003029, ...	Numérique, PK
numero_carte_etudiant	CHAR(8)	NN	22102591	Numérique
est_redoublant	BOOLEAN	NN	Oui/Non	
est_boursier	BOOLEAN	NN	Oui/Non	

Enseignants	Type	Null/Non null	Exemple	Contraintes
id_persn_enseignant	CHAR(9)	NN	122003861, ...	Numérique, PK
poste	VARCHAR(20)	NN	Enseignant, chef de département	Alphabétique
specialite	VARCHAR(20)	NN	Informatique	Alphabétique

Directeurs	Type	Null/Non null	Exemple	Contraintes
id_persn_ directeur	CHAR(9)	NN	122004921, …	Numérique, PK
titre_poste	VARCHAR(20)	NN	Directeur du pôle informatique, …	Alphabétique
date_ nomination	DATE	NN	2024-05-20	aaaa-mm-dd
duree_mandat	INT	NN	5 ans	duree_mandat > 0

Cours	Type	Null/Non null	Exemple	Contraintes
id_cours	CHAR(8)	NN	5BEPRO4D, ...	Alphanumérique, PK
nom_cours	VARCHAR(40)	NN	Proba Stat	Alphabétique
heure_debut	TIME	NN	8:30	
heure_fin	TIME	NN	11:45	> heure_debut
date	DATE	NN	2024-09-16	aaaa-mm-dd

Départements	Type	Null/Non null	Exemple	Contraintes
id_departeme nt	CHAR(7)	NN	CYI9500	Alphanumérique ,PK
nom	VARCHAR(80)	NN	Informatique	Alphabétique
adresse	VARCHAR(10 0)	NN	Saint Martin 95000	Alphanumérique

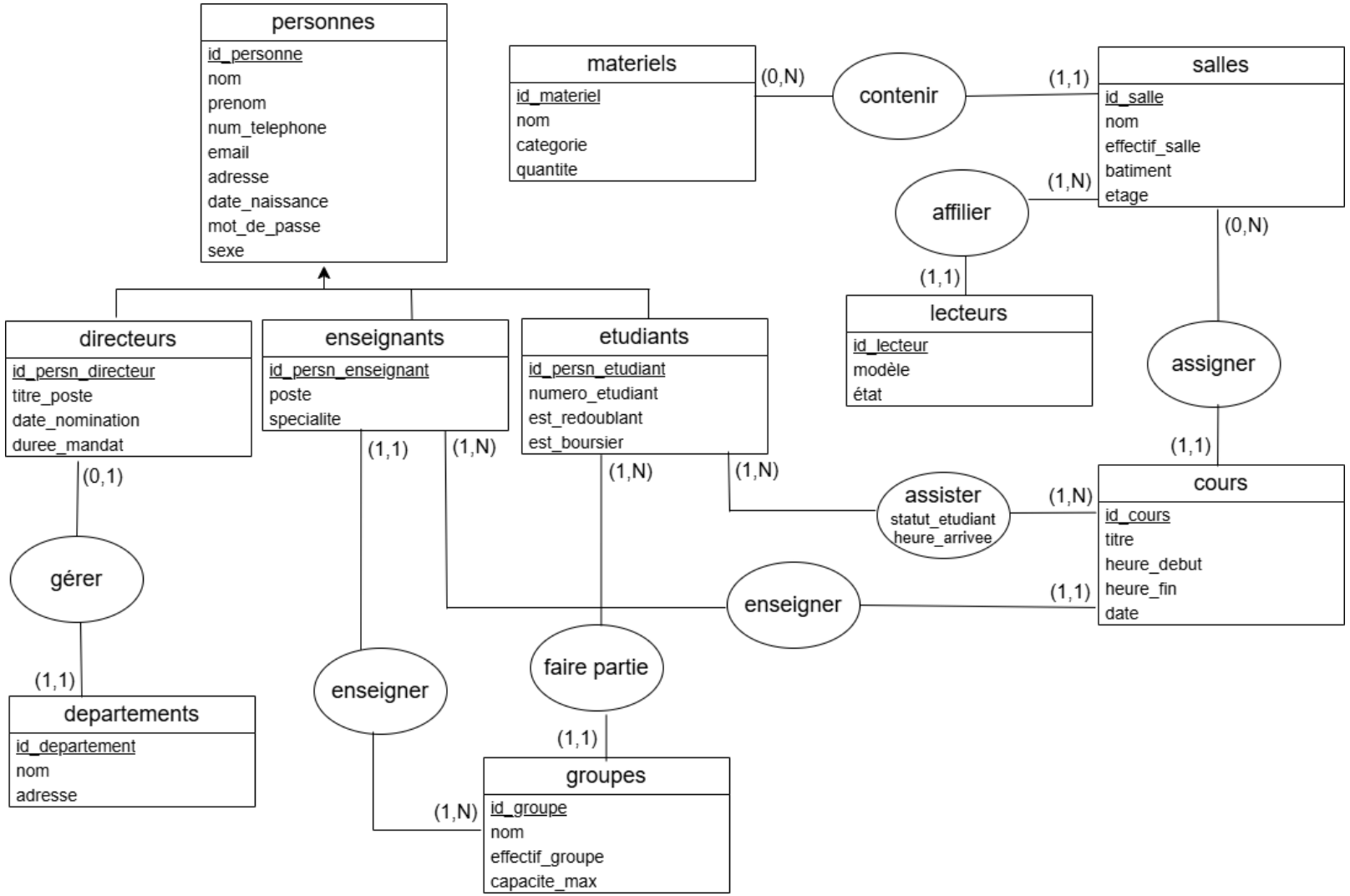
Materiels	Type	Null/Non null	Exemple	Contraintes
id_materiel	CHAR(8)	NN	M1000200	Alphnumérique, PK
nom	VARCHAR(40)	NN	Informatique	Alphabétique
categorie	ENUM	NN	Materiel informatique, materiel de labo	Numérique
quantite	INT	NN	12 exemplaires	

Lecteurs	Type	Null/Non null	Exemple	Contraintes
id_lecteur	CHAR(8)	NN	156A	Alphanumérique, PK
modele	VARCHAR(45)	NN	MBK78	Alphanumérique
etat	ENUM	NN	fonctionnel	Alphabétique

Groupes	Type	Null/Non null	Exemple	Contraintes
id_groupe	CHAR(9)	NN	grpAinfo	Alphanumérique, PK
capacite_max	INT	NN	25	numérique
nom	VARCHAR(20)	NN	groupe A	Alphabétique
effectif_groupe	INT	NN	20 etudiants	0 < effectif_groupe < capacite_max

Salles	Type	Null/Non null	Exemple	Contraintes
id_salle	CHAR(7)	NN	info2024A	Alphabétique, PK
nom	VARCHAR(10)	NN	Salle A2	Alphabétique
effectif_salle	INT	NN	150	Numérique, 10 < effectif_salle < 200
bâtiment	VARCHAR(10)	NN	Batiment A	Alphabétique
etage	INT	NN	5	Numérique

5. Schéma E/A


personnes
id_personne
nom
prenom
num_telephone
email
adresse
date_naissance
mot_de_passe
sexe
materiels
id_materiel
nom
categorie
quantite
(0,N)
contenir
(1,1)
salles
id_salle
nom
effectif_salle
batiment
etage
(1,N)
affilier
(1,1)
lecteurs
id_lecteur
modèle
état
(0,N)
assigner
(1,1)
directeurs
id_persn_directeur
titre_poste
date_nomination
duree_mandat
(0,1)
gérer
(1,1)
departements
id_departement
nom
adresse
enseignants
id_persn_enseignant
poste
specialite
(1,1)
(1,N)
enseigner
etudiants
id_persn_etudiant
numero_etudiant
est_redoublant
est_boursier
(1,N)
(1,N)
assister
statut_etudiant
heure_arrivee
(1,N)
cours
id_cours
titre
heure_debut
heure_fin
date
enseigner
(1,1)
faire partie
(1,1)
groupes
id_groupe
nom
effectif_groupe
capacite_max
(1,N)

6. Schéma Relationnel

Personnes (**id_personne**, nom, prenom, num_telephone, email, adresse, date_naissance, mot_de_passe, sexe)

Directeurs (**#id_persn_directeur**, titre_poste, date_nomination, duree_mandat)

Etudiants (**#id_persn_etudiant**, numero_carte_etudiant, est_redoublant, est_boursier)

Materiels (**id_materiel**, nom, categorie, quantite)

Groupes (**id_groupe**, nom, capacite_max, effectif_groupe, *#id_etudiant*)

Departements (**id_departement**, nom, adresse)

Enseignants (**#id_persn_enseignant**, poste, specialite, *#id_groupe*)

Salles (**id_salle**, nom, effectif_salle, batiment, etage, *#id_materiel*)

Lecteurs (**id_lecteur**, modèle, état, *#id_salle*)

Cours (**id_cours**, titre, heure_debut, heure_fin, date, *#id_salle*, *#id_enseignant*)

Assister (**#id_etudiant**, **#id_cours**, statut_etudiant, heure_arrivee)

7. Jeu de données

Personnes

id_ personnes	nom	prenom	num_ telephone	email	adresse	mot_de_ passe	date_ naissance	genre
122003861	MEBARKI	Khalifa	0768760148	khalifa.m@ gmail.com	23 bd du port Cergy	ndskaeh3003	2001-09-23	Homme
122003862	DUPONT	Marie	0751234567	marie.dup@ gmail.com	15 rue du Lac Paris	test2002	1998-04-12	Femme
122003863	LECLERC	Tom	0749876543	thomas.l@ gmail.com	10 av des Champs Paris	3U3UNN	1995-07-30	Homme
122003864	LEMAIRE	Sophie	0765432178	sophie.le@ gmail.com	32 rue des Fleurs Lyon	mdppdm	2000-03-15	Femme
122003865	NGUYEN	Minh	0720147893	minh.ng@ gmail.com	7 impasse du Moulin	mbk6nan	1993-02-25	Homme

Etudiants

id_persn_etudiant	numero_carte_etudiant	est_redoublant	est_boursier
122003861	22102591	Non	Oui
122003862	22102592	Oui	Non
122003863	22102593	Non	Oui
122003864	22102594	Non	Non
122003865	22102595	Oui	Oui

Enseignants

id_persn_ enseignant	poste	specialite	id_groupe
122003866	Professeur informatique	Informatique	G1
122003867	Chef de département	Mathématiques	G2
122003868	Enseignant	Physique	G3
122003869	Enseignant	Chimie	G3

Directeurs

id_persn_directeur	titre_poste	date_nomination	duree_mandat
950308999	Directeur pôle informatique	2024-05-20	3
950308998	Directeur pôle chimie	2027-05-12	3
950308997	Directeur pôle physique	2022-11-03	3
950308996	Directeur pôle scientifique	2024-06-19	5
950308990	Directeur pôle littéraire	2023-04-28	3

Cours

id_cours	titre	heure_debut	heure_fin	date	fk_salle	fk_enseign
5BEPRO4D	Proba Stat	08:30	11:45	2024-09-16	info2024A	122003868
5BEMAT2A	Algèbre	10:00	12:30	2024-09-17	info2024E	122003869
5BEINF3B	Informatique	13:00	15:30	2024-09-18	info2024C	122003861
5BEPHY1C	Physique	09:00	11:00	2024-09-19	info2024C	122003863
5BECHI2D	Chimie	14:00	16:30	2024-09-20	info2024D	122003864

Départements

id_departement	nom	adresse
CY19500	Informatique	Saint Martin 95000
CY19501	Mathématiques	15 Rue de la Paix
CY19502	Physique	10 Avenue des Sciences
CY19503	Chimie	7 Place du Marché

Materiels

id_materiel	nom	categorie	quantite
M1000200	PC Fixe	Materiel informatique	20
M2000400	Bécher	Laboratoire	12
M3000100	Multiprise	Toutes catégories	5

Lecteurs

id_lecteur	modele	etat	fk_salle
156A	MBK78	fonctionnel	info2024A
157B	MBK79	en panne	info2024B
158C	MBK80	fonctionnel	info2024C
150E	MBK81	fonctionnel	info2024E

Salles

id_salle	nom	effectif_salle	batiment	etage	id_materiel
info2024A	Salle A2	150	batiment A	5	M1000200
info2025B	Salle B3	100	batiment B	3	M3000100
info2026C	Salle C1	90	batiment C	2	M1000200
info2027D	Salle D4	120	batiment D	4	M3000100

Groupes

id_groupe	capacite_max	nom
G1	25	groupe A
G2	30	groupe B
G3	28	groupe C
G4	20	groupe D

Assister

id_etudiant	id_cours	statut_etudiant	heure_arrivee
122003861	5BEPRO4D	présent	8:28
122003862	5BEPRO4D	non présent	null
122003863	5BEINF3B	présent	13:52
122003864	5BEINF3B	non présent	null

8. Échanges réseaux / Diagrammes Applicatifs

Échanges réseaux

Dans le cadre de notre projet de gestion des présences, plusieurs types d'échanges réseau seront mis en place entre les différentes parties prenantes (enseignants, étudiants) et la base de données centrale. Par exemple, lorsqu'un étudiant scanne son badge sur un lecteur, les informations (comme l'ID du badge et l'heure) seront transmises à la base de données pour y être stockées et associées à l'étudiant concerné. Les enseignants pourront accéder à cette base via une interface pour consulter et valider les présences, tandis que les étudiants pourront également consulter leur historique de présence. Tous ces échanges seront sécurisés et optimisés pour garantir l'intégrité des données dans la base.

Enregistrement de la présence des étudiants

Les étudiants utiliseront leur carte étudiante pour marquer leur présence en passant devant un lecteur de carte à l'entrée de la salle. Chaque passage enverra une requête au serveur pour enregistrer leur présence dans le système. Le serveur recevra les informations (comme l'identifiant de l'étudiant) et les stockera dans la base de données.

Vérification et prévention de la duplication des présences

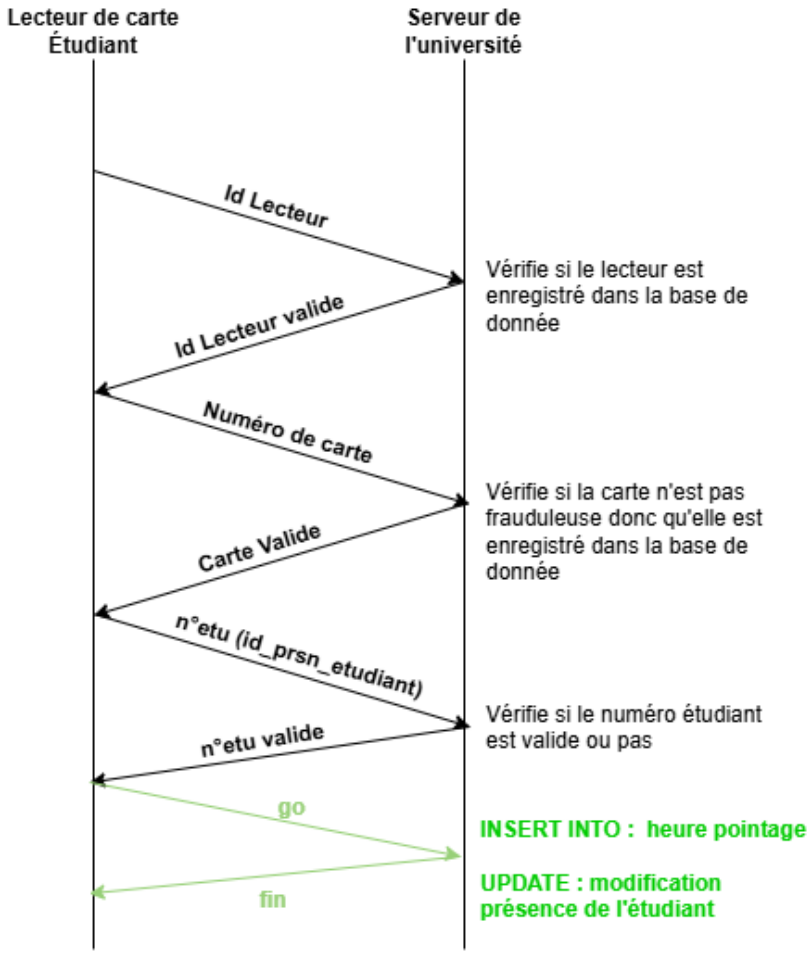
Lorsque l'étudiant passe sa carte devant le lecteur pour enregistrer sa présence, le serveur vérifie d'abord si l'étudiant est déjà noté comme présent pour ce cours. Si l'étudiant est déjà enregistré (soit automatiquement via le lecteur de carte, soit manuellement par l'enseignant), aucune mise à jour ne sera effectuée dans la base de données, évitant ainsi les duplications d'enregistrements de présence. Cette vérification permet de garantir que chaque étudiant ne sera compté qu'une seule fois par cours.

Diagrammes Applicatifs

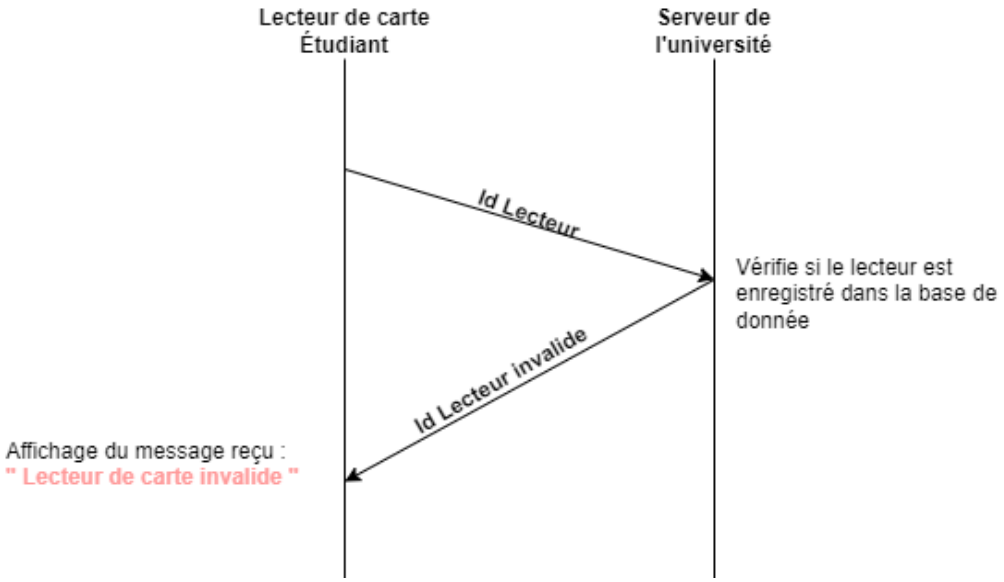
1) Diagramme applicatif : scénario accepté

Les numéros d'étudiants sont déjà enregistrés mais non présents, par contre si une mise à jour est effectuée (c'est-à-dire si l'étudiant badge sa carte sur le lecteur de présence), leur présence est validée tout en ajoutant l'heure de pointage à cette mise à jour (Elle est insérée [INSERT] dans la base de données et le statut de l'étudiant passera de "absent" à "présent" [UPDATE]).

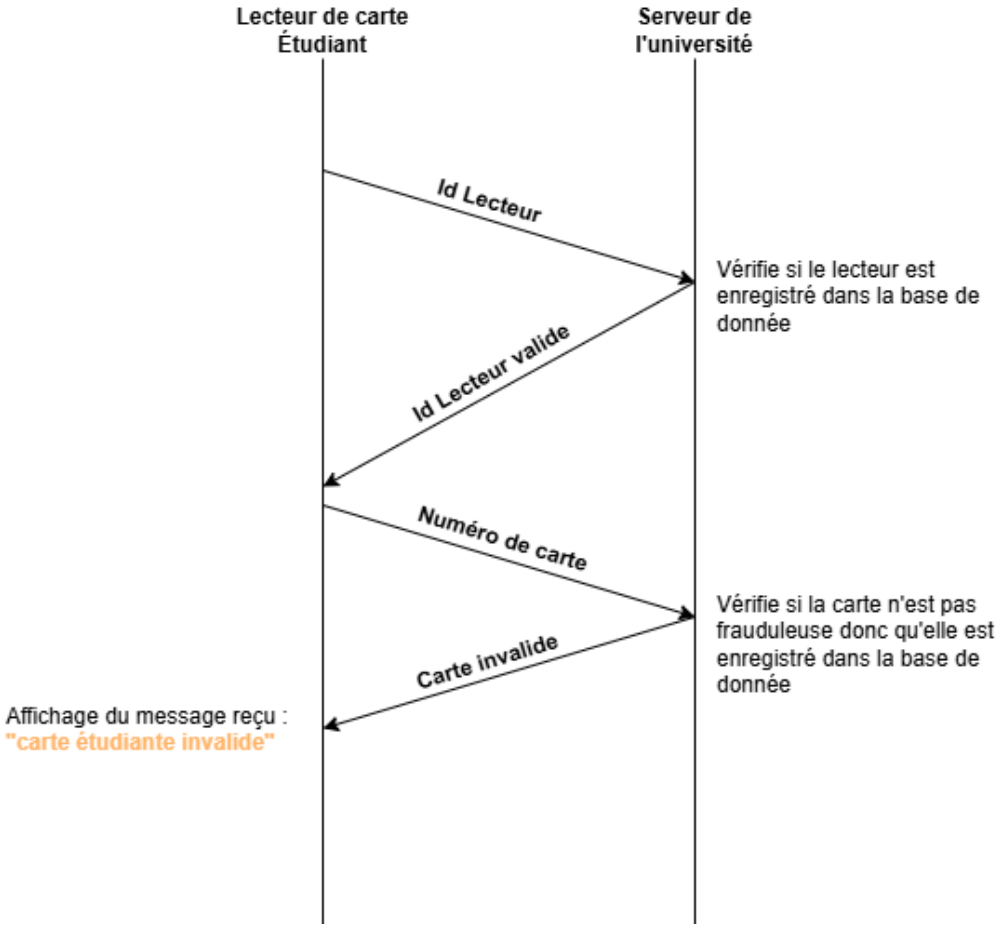
Protocole : TCP | N° de port : 12345



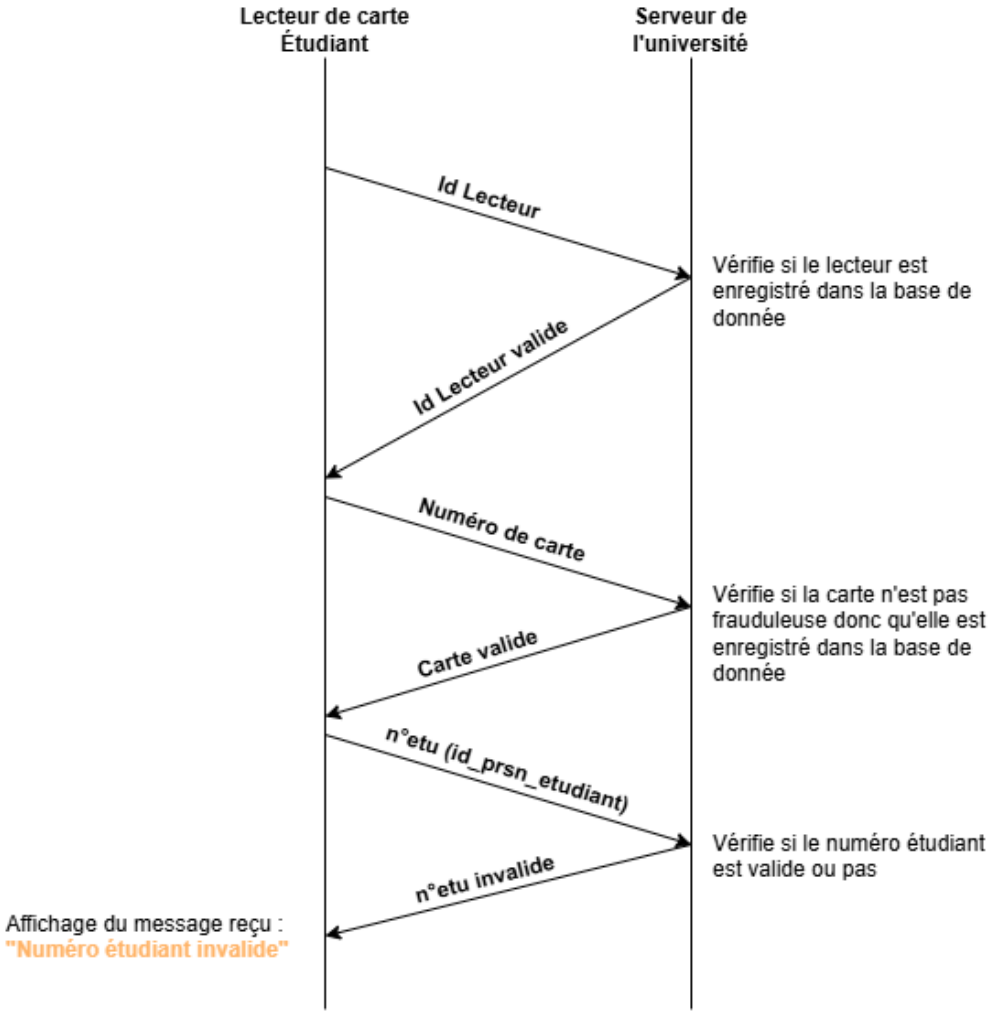
2) Diagramme applicatif : scénario ID lecteur invalide



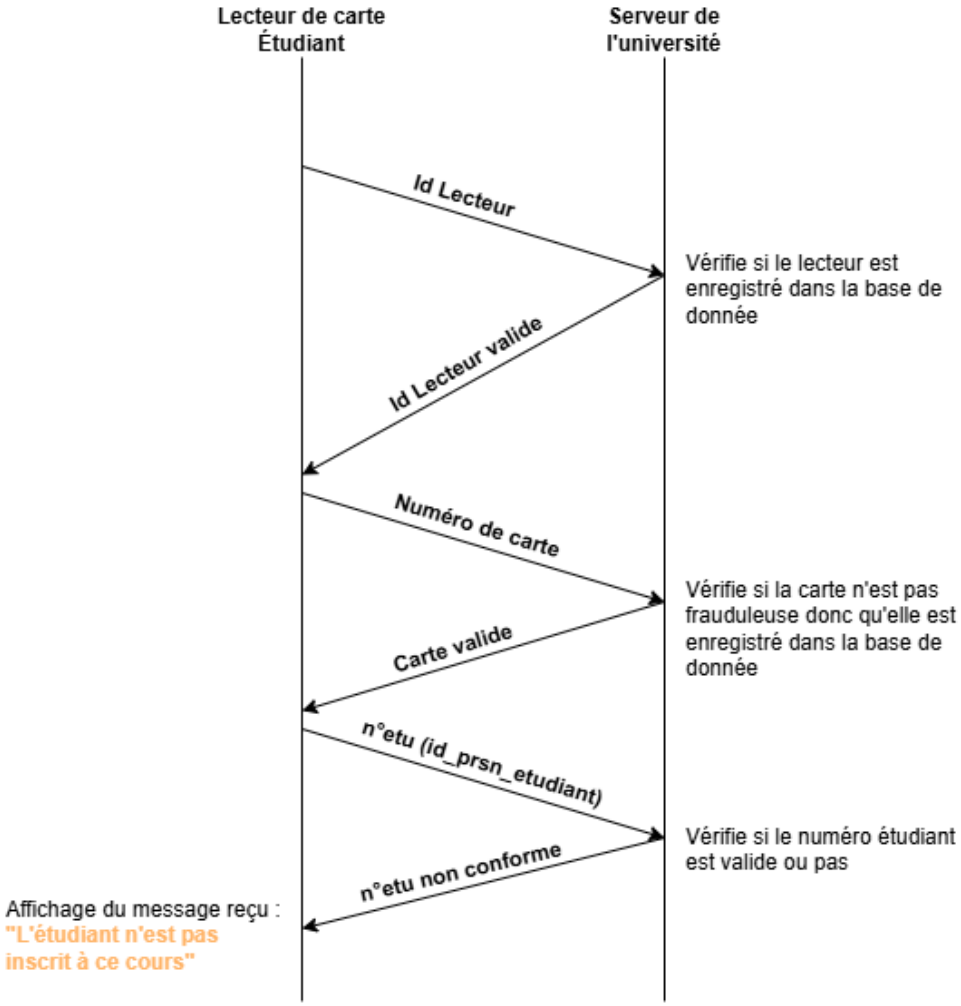
3) Diagramme applicatif : scénario ID carte étudiant invalide



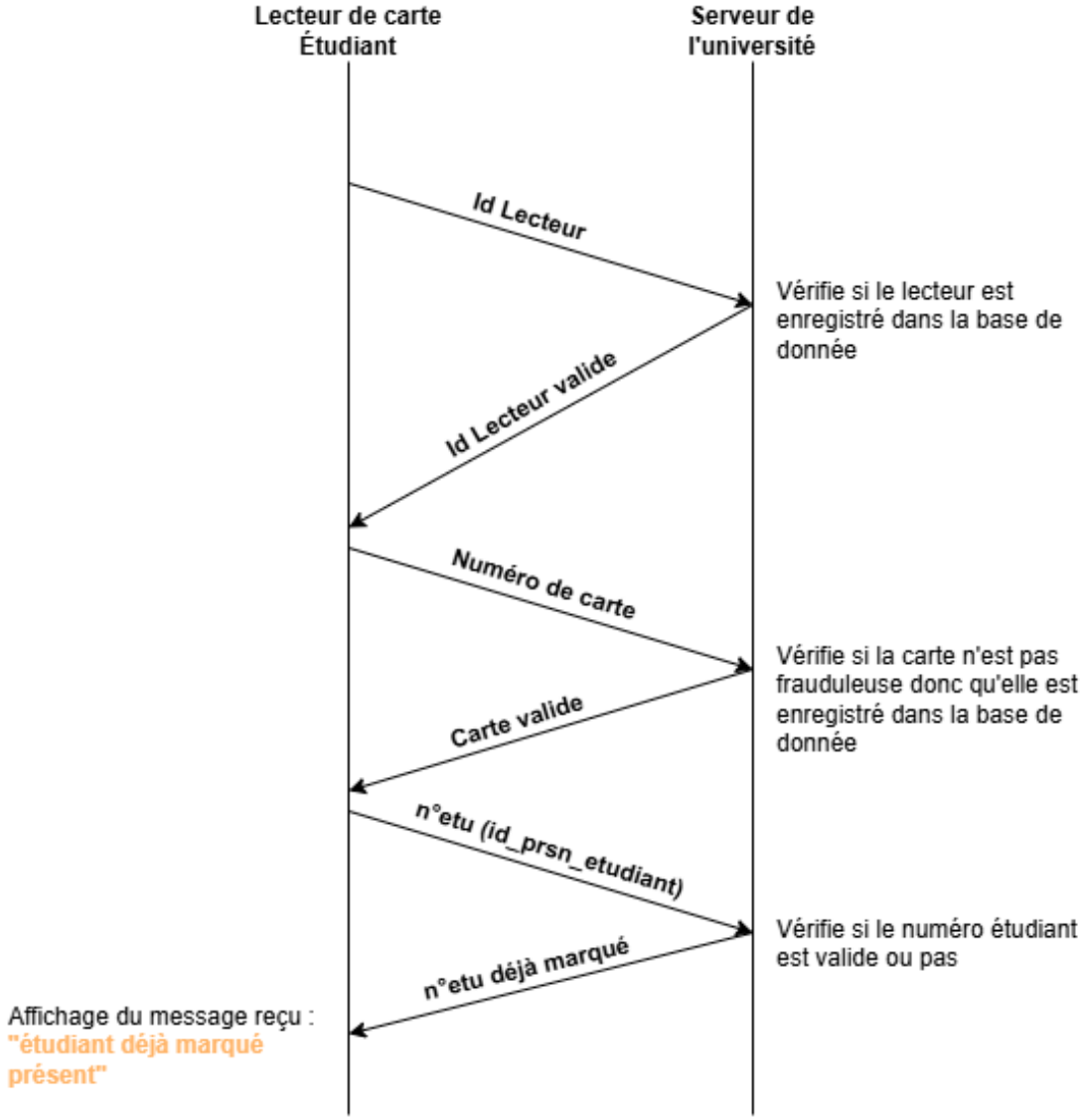
4) Diagramme applicatif : scénario n°étudiant invalide



5) Diagramme applicatif : scénario l'étudiant n'est pas inscrit à ce cours



6) Diagramme applicatif : scénario l'étudiant badge deux fois



Test des échanges TCP avec ncat

Test du serveur avec un ncat client :

On note tout d'abord la commande avec l'adresse ip et le port :

```
C:\Users\lenovo i5>ncat localhost 12345
```

Et en allant sur le terminal serveur, on retrouve bien que le client a réussi à se connecter :

```
[Running] python -u "c:\Users\lenovo i5\L3 - BDD\serveur_pythonn.py"
2024-10-27 23:35:35,985 - INFO - Serveur en ecoute sur 127.0.0.1:12345...
2024-10-27 23:35:36,043 - INFO - Connexion à la base de données PostgreSQL établie.
2024-10-27 23:35:43,596 - INFO - Client connecté : ('127.0.0.1', 19365)
```

Ensuite, on note la commande pour le lecteur :

```
LECTEUR:156A
ID Lecteur valide
```

Sur le serveur, on retrouve bien que les données ont été reçues. Le serveur renvoie ensuite "ID Lecteur valide" (cf. capture ci-dessus) :

```
2024-11-23 19:59:28,016 - DEBUG - Message reçu de ('127.0.0.1',
23025) : LECTEUR:156A
```

On note après cela le numéro de carte étudiant :

```
CARTE:22102592
Carte valide
```

Sur le serveur, on retrouve bien que les données ont été reçues. Le serveur renvoie ensuite "Carte valide" (cf. capture ci-dessus) :

```
2024-11-23 20:00:34,444 - DEBUG - Message reçu de ('127.0.0.1',
23025) : CARTE:22102592
```

Enfin, on note le numéro étudiant (correspondant à id_persn_etudiant) sur ncat

```
ETUDIANT:122001186
Numero etudiant valide
Presence enregistree
```

:

Sur le serveur, on retrouve bien que les données ont été reçues. Le serveur renvoie ensuite "N°étudiant valide" et "Présence enregistrée" (cf. capture ci-dessus) :

```
2024-11-23 20:01:11,578 - DEBUG - Message reçu de ('127.0.0.1',
23025) : ETUDIANT:122001186
```

On vérifie donc ensuite sur la BDD si l'étudiant est bien enregistrée et c'est bien le cas :

Actions		id_cours	id_persn_etudiant	statut_etudiant	heure_arriver
Éditer	Effacer	5BEPRO4D	122001185	present	08:25:00
Éditer	Effacer	5BEPRO4D	122001187	absent	*NULL*
Éditer	Effacer	5BEPRO4F	122001188	absent	*NULL*
Éditer	Effacer	5BEPRO4F	122001189	absent	*NULL*
Éditer	Effacer	5BEPRO4E	122001186	true	18:59:26.422135

L'affichage final obtenu sur le ncat client :

```
LECTEUR:156A
ID Lecteur valide
CARTE:22102592
Carte valide
[redacted]

ETUDIANT:122001186
Numero etudiant valide
Presence enregistree
^C
C:\Windows\system32>
```

Test du client avec un ncat serveur:

On note tout d'abord la commande avec le n° port pour lancer le ncat en mode serveur :

```
C:\Windows\system32>ncat -l 12345
```

Puis on lance notre client et on note l'ID du lecteur sur le terminal :

```
Connecté au serveur sur 127.0.0.1:12345
Entrez l'ID du lecteur : 156A
```

La console affiche bien l'ID du lecteur envoyé par le client :

```
C:\Windows\system32>ncat -l 12345
LECTEUR:156A
```

Nous saisissons le message que le serveur va renvoyer au client "ID lecteur valide".

```
LECTEUR:156A
ID Lecteur valide
```

Et le client reçoit bien le message du serveur :

```
Réponse reçue du serveur : ID Lecteur valide
Le lecteur est valide.
```

Ensuite, on note le numéro de carte côté client et le ncat serveur le reçoit bien, on note donc le message que le serveur va renvoyer au client “Carte valide” :

```
CARTE:22102597
Carte valide
```

Le client reçoit bien le message et passe à l’étape suivante “Entrez le n°étudiant” :

```
La carte est valide.
Entrez le numéro étudiant : 122001187
```

Après avoir noté le n°étudiant côté client, le serveur le reçoit et on note ensuite le message renvoyé par le serveur “Numéro étudiant valide” et “Présence enregistrée” :

```
ETUDIANT:122001187
Numero etudiant valide
Presence enregistree
```

Le client reçoit bien ces informations et conclu la connexion :

```
L'étudiant est valide.
Présence de l'étudiant enregistrée avec succès.
Connexion terminée.
```

L’affichage final obtenu sur le ncat serveur et le client :

```
C:\Windows\system32>ncat -l 12345
LECTEUR:156A
ID Lecteur valide
CARTE:22102597
Carte valide
ETUDIANT:122001187
Numero etudiant valide
Presence enregistree
```

```
Connecté au serveur sur 127.0.0.1:12345
Entrez l'ID du lecteur : 156A
Réponse reçue du serveur : ID Lecteur valide
Le lecteur est valide.
Entrez le numéro de carte : 22102597
La carte est valide.
Entrez le numéro étudiant : 122001187
L'étudiant est valide.
Présence de l'étudiant enregistrée avec succès.
Connexion terminée.
```

9. Connexion à la BDD

Pour la partie réseau, la connexion à la base de données se configure au niveau du serveur codé en python. Cependant, par soucis de sécurité, les codes de connexion ne sont pas directement intégrés au serveur python mais dans un fichier de configuration “config.ini”.

Ce fichier, contenant les “logs” de la base de données, est appelé par le serveur python pour récupérer ces fameux codes de connexion

```
[server]
host = 127.0.0.1
port = 12345

[database]
db_host =  postgresql-mebarki.alwaysdata.net
db_name = mebarki_dbetu
db_user = mebarki
db_password = phpPgAdmin

```

:

Pour ceux qui est de la partie web, le même principe y est appliqué : Les codes de connexion ne sont pas implémentés dans les différents codes PHP du site mais bien dans un seul fichier “config.inc.php” pour plus de sécurité :

```php
<?php
// Fichier de configuration
// Paramètres de connexion à la base de données
define('DB_HOST', 'postgresql-mebarki.alwaysdata.net');
define('DB_PORT' , '5432');
define('DB_NAME', 'mebarki_dbetu');
define('DB_USER', 'mebarki');
define('DB_PASSWORD', 'phpPgAdmin');

$db = new
PDO("pgsql:host=".DB_HOST.";port=".DB_PORT.";dbname=".DB_NAME.";user=".DB_USER.";password="
.DB_PASSWORD);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// Démarrage de la session
session_start();
?>
```

10.Les différentes requêtes

Partie Réseau

Les différentes requêtes vers la base de données de la partie réseau se trouve du côté serveur en python, nous les avons disposées dans des fonctions pour les organiser intelligemment comme ceci :

- La vérification du lecteur valide :

```
SELECT * FROM lecteurs WHERE id_lecteur = id_lecteur
```

- La vérification de la carte étudiante :

```
SELECT * FROM etudiants WHERE numero_carte_etudiant = numero_carte_etudiant
```

- La vérification du numéro étudiant :

```
SELECT * FROM etudiants WHERE id_persn_etudiant = id_persn_etudiant
```

- Et enfin, si toute les vérifications sont validées, l'enregistrement de l'étudiant :

```
SELECT statut_etudiant FROM assister WHERE numero_carte_etudiant = numero_carte_etudiant
```

Si l'étudiant n'a pas encore pointé sa présence : On insère l'heure de passage + on met à jour le statut de sa présence :

```
INSERT INTO assister (numero_carte_etudiant, heure_arriver) VALUES (numero_carte_etudiant, NOW() )

UPDATE assister SET statut_etudiant = TRUE WHERE numero_carte_etudiant = numero_carte_etudiant
```

Note : Si une vérification n'est pas valide (cf. Les diagrammes applicatifs) la fonction renvoie "Faux" et le code serveur exécute des actions en conséquence (envoie de l'information au client, …) [Cf. Annexes pour le cœur des codes] .

Partie Web

Les différentes requêtes vers la base de données de la partie web se trouvent dans les différents codes PHP du site comme par exemple la page d'actualités ou de connexion. Nous les avons donc organisées intelligemment comme ceci :

- Récupère le nombre d'étudiants de l'université :

```
SELECT COUNT(*) AS nombre_etudiant FROM etudiants
```

- Récupère les matériels disponible (triés par quantité) :

```
SELECT nom_materiel, quantite FROM materiels ORDER BY quantite ASC
```

- Récupère la moyenne d'effectifs de toutes les salles :

```
SELECT ROUND(AVG(effectif_salle), 2) AS moyenne_salle FROM salles
```

- Récupère la liste des salles ayant un effectif supérieur à la moyenne :

```
SELECT id_salle, effectif_salle FROM salles WHERE effectif_salle > ( SELECT
ROUND(AVG(effectif_salle), 2) FROM salles ) ORDER BY effectif_salle ASC
```

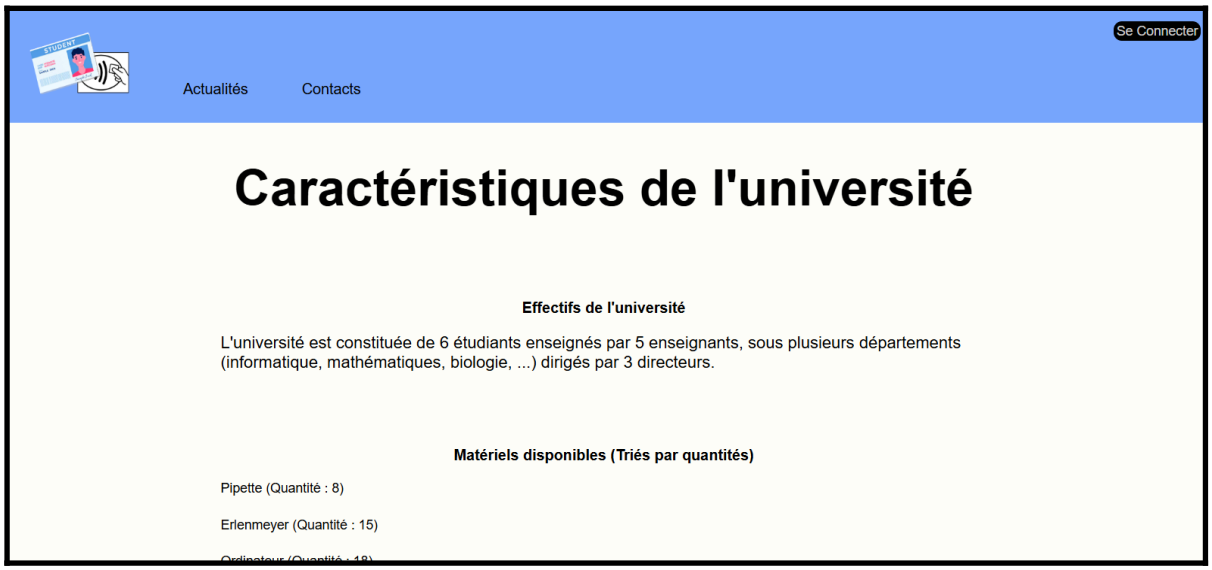
- Récupère la personne dont l'email et le mot de passe correspond :

```
SELECT * FROM personnes WHERE email = :email AND mot_de_passe
= :mot_de_passe
```

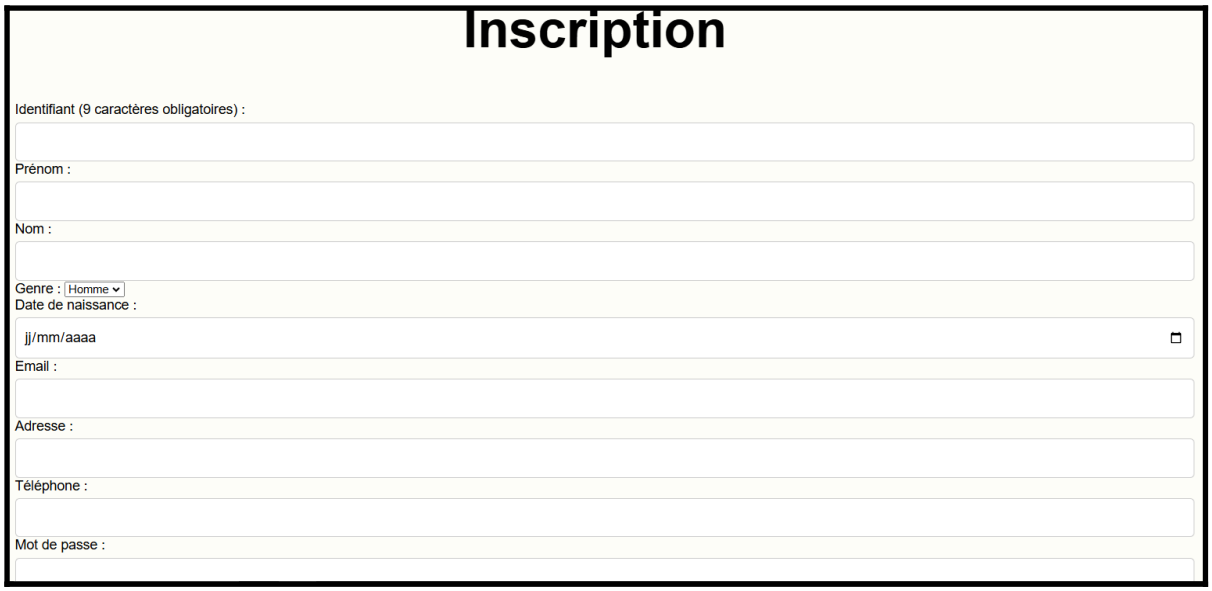
11. Site Web

Pour ce qui est de notre site web, nous avons différentes pages utiles à notre projet :

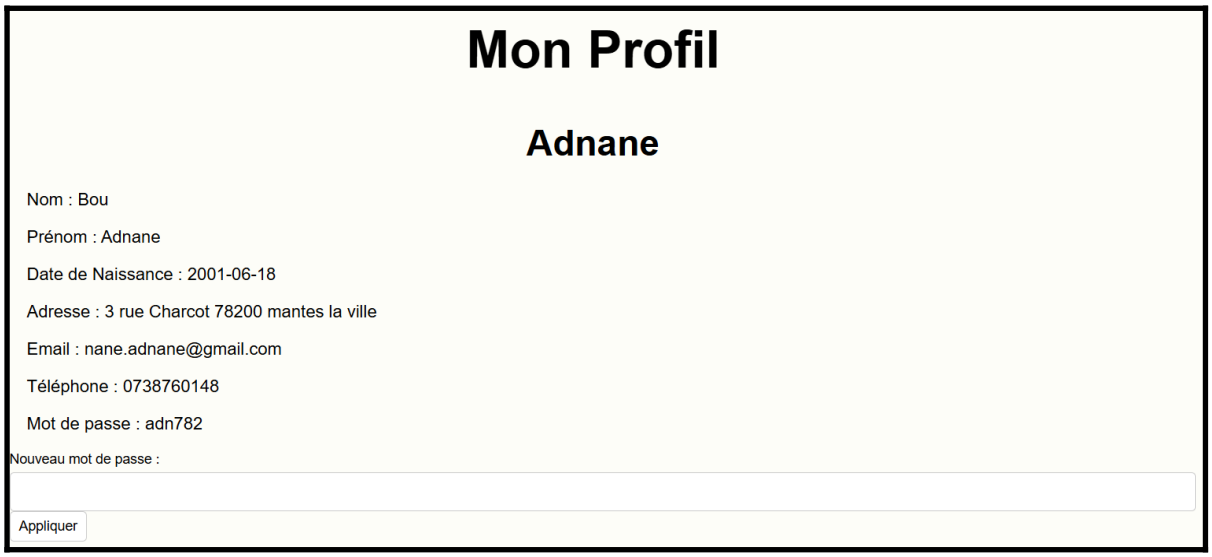
- La page “Actualités” contenant une multitude d’informations concernant l’université (nombres d'étudiants, de salles, matériels disponibles,…).



- La page “S’inscrire” qui permet à un utilisateur d’avoir un compte :



- La page "Mon Profil" (qui n'apparaît que si l'utilisateur est connecté) affiche les informations de l'utilisateur et offre également la possibilité de modifier son mot de passe.



L'URL de notre site : https://mebarki.alwaysdata.net/

12. Conclusion et remerciement

En conclusion, durant les semaines passées à travailler sur ce projet, nous avons su mettre à profit nos compétences et les combiner afin d'aboutir à un rendu satisfaisant les objectifs que nous nous étions fixés.
Nous avons dû faire face à de nombreuses contraintes, notamment celle du temps, mais nous avons pu rebondir et fournir un projet complet et innovant.
Ce projet nous a également permis d'étendre nos perspectives et connaissances sur les bases de données et les échanges réseaux, ce qui nous a permis d'acquérir des compétences solides dans ces domaines.

Enfin, nous souhaitons remercier chaleureusement les différents professeurs qui nous ont accompagnés lors de ce projet :

- **JEN Tao-Yuan**, notre professeur de bases de données, qui nous a enseigné tout l'aspect modélisation des bases de données lors de ses cours magistraux.
- **DANG NGOC TUYET Tram**, pour nous avoir appris les bases des réseaux que nous avons utilisées au cours de ce projet.
- Et enfin, **LEMAIRE Marc**, qui nous a accompagnés tout au long du déroulement de ce projet.

Annexes

Coeurs des codes réseaux (Client/Serveur)

Partie Client (JAVA):

Premièrement, une connexion est établie avec le serveur en y ajoutant un timeout si cela prend trop de temps.

```
        // Connexion au serveur avec un timeout
        socket.connect(new InetSocketAddress(serverHostname, serverPort), timeout);
        System.out.println("Connecté au serveur sur " + serverHostname + ":" + serverPort);

        // Configuration du timeout de lecture
        socket.setSoTimeout(timeout);
```

Ensuite, un flux est ouvert pour pouvoir envoyer/recevoir les données et on commence par envoyer les premières données (ici l'ID du lecteur).

```
        // Flux pour envoyer et recevoir des données
        PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);
        BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        Scanner scanner = new Scanner(System.in);
        // Étape 1 : Envoi de l'ID du lecteur
        System.out.print("Entrez l'ID du lecteur : ");
        String idLecteur = scanner.nextLine();
        writer.println("LECTEUR:" + idLecteur);
        // Vérifier la réponse du serveur pour l'ID du lecteur
        String response = reader.readLine();
        System.out.println("Réponse reçue du serveur : " + response);
        if (!"ID Lecteur valide".equals(response)) {
            System.out.println("Erreur : " + response);
            return; // Stopper le client si l'ID lecteur est invalide
        }
        System.out.println("Le lecteur est valide.");
```

Puis, on vérifie la réponse du serveur par rapport à ce qui a été envoyé et on obtient un résultat positif ou négatif selon le reçu. Cette partie du code est donc répétée pour les différents échanges avec le serveur (Carte, N°Etudiant, …).

Par contre, si une donnée est invalide, le code est stoppé et une exception est levé, voici les exceptions du code java :

```
} catch (SocketTimeoutException e) {
        System.err.println("Erreur : Le serveur n'a pas répondu dans le délai imparti.");
    } catch (UnknownHostException e) {
        System.err.println("Erreur : Hôte inconnu - " + serverHostname);
    } catch (ConnectException e) {
        System.err.println("Erreur : Connexion refusée sur " + serverHostname + ":" + serverPort);
    } catch (IOException e) {
        System.err.println("Erreur d'E/S lors de la connexion à " + serverHostname + ":" + serverPort);
    }
```

Partie Serveur (PYTHON) :

Pour la partie serveur, on commence par se connecter à la base de données par le biais d'un fichier de configuration :

```
# Charger les paramètres de la base de données depuis le fichier de configuration
DB_HOST = config.get('database', 'db_host')
DB_NAME = config.get('database', 'db_name')
DB_USER = config.get('database', 'db_user')
DB_PASSWORD = config.get('database', 'db_password')
# Paramètres du serveur (configurables)
HOST = '127.0.0.1'
PORT = 12345
# Connexion à la base de données PostgreSQL
def connect_to_database():
    try:
        # Connexion PostgreSQL
        conn = psycopg2.connect(
            host=DB_HOST,
            database=DB_NAME,
            user=DB_USER,
            password=DB_PASSWORD
        )
        logging.info("Connexion a la base de donnees PostgreSQL etablie.")
        return conn
    except psycopg2.Error as e:
        logging.error(f"Erreur de connexion a la base de donnees : {e}")
        sys.exit(1)
```

Puis, plusieurs fonctions sont utilisées pour les requêtes à la base de données (cf. Les différentes requêtes) au niveau du cœur du code. Voici comment ce présente ces actions :

```
            # Étape 1: Vérification du lecteur
            if message.startswith("LECTEUR"):
                id_lecteur = message.split(":")[1].strip()
                if verifier_lecteur(db_conn, id_lecteur):
                    conn.sendall(b"ID Lecteur valide\n")
                else:
                    conn.sendall(b"Lecteur de carte invalide\n")
                    return
            # Étape 2: Vérification de la carte
            elif message.startswith("CARTE"):
                numero_carte = message.split(":")[1].strip()
                if verifier_carte(db_conn, numero_carte):
                    conn.sendall(b"Carte valide\n")
                else:
                    conn.sendall(b"Carte invalide\n")
                    return
            # Étape 3: Vérification du numéro étudiant
```

```
        elif message.startswith("ETUDIANT"):
            numero_etudiant = message.split(":")[1].strip()
            if verifier_etudiant(db_conn, numero_etudiant):
                conn.sendall(b"Numero etudiant valide\n")

                # Étape 4: Essayer d'enregistrer la présence
                presence_enregistree = enregistrer_presence(db_conn, numero_etudiant)
                if presence_enregistree == "non enregistree":
                    conn.sendall(b"Presence enregistree\n")
                elif presence_enregistree == "deja enregistree":
                    conn.sendall(b"Presence deja enregistree\n")
                else :
                    conn.sendall(b"Etudiant inconnu\n")
            else:
                conn.sendall(b"Numero etudiant invalide\n")

        # Sinon, message inconnu
        else:
            conn.sendall(b"Commande inconnue\n")
```

Avant de faire ces aspects de programmation, le serveur vérifie le contenu des messages envoyés par le client (taille, possibles injection SQL, …) :

```
        data = conn.recv(4096)

        # Déconnexion client
        if not data:
            logging.info(f"Deconnexion du client {addr}")
            break

        # Vérification de la taille et contenu des données
        if len(data) > 1024:
            logging.warning(f"Depassement de taille de buffer de {addr}")
            conn.sendall(b"Erreur: taille de buffer depassee.\n")
            continue

        # Décodage et nettoyage pour éviter injection SQL
        message = data.decode('utf-8').strip()
        logging.debug(f"Message recu de {addr} : {message}")
```

Coeurs des codes Web (PHP)

Exemple d'affichage d'une donnée venant de la base de données :

```php
<?php
    $query = "SELECT ROUND(AVG(effectif_salle), 2) AS moyenne_salle FROM salles";
    $stmt = $db->query($query);
    $result = $stmt->fetch(PDO::FETCH_ASSOC);
    $moyenne_salle = $result['moyenne_salle'];

    echo "<p>L'effectif moyen de toutes les salles de l'université est de $moyenne_salle personnes.</p>";
?>
```

Le formulaire de connexion pour se connecter + l'envoi de ces données à la bdd :

```html
<form method="POST">
            <div class="input-group">
                <label for="email">Identifiant (mail)</label>
                <input type="text" id="email" name="email" maxlength="64" required>
            </div>
            <div class="input-group">
                <label for="mot_de_passe">Mot de passe</label>
                <div class="password-input">
                    <input type="password" id="mot_de_passe" name="mot_de_passe" maxlength="70"
required>
                </div>
            </div>
            <input type="submit" value="Se connecter">
</form>
```

```php
// Requête SQL pour vérifier les identifiants
    $query = "SELECT * FROM personnes WHERE email = :email AND mot_de_passe
= :mot_de_passe";
    $stmt = $db->prepare($query);
    $stmt->bindParam(':email', $email);
    $stmt->bindParam(':mot_de_passe', $mot_de_passe);
    $stmt->execute();
    // Vérification des résultats de la requête
    $user = $stmt->fetch(PDO::FETCH_ASSOC); #
    if ($user) {
        // Redirection vers la page d'accueil si les identifiants sont corrects
        // Stockez l'identifiant de l'utilisateur dans la session
        $_SESSION['user_id'] = $user['id_personne'];
        header('Location: index.php');
        exit();
```

Le code qui permet la modification du mot de passe de l'utilisateur :

```php
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['mot_de_passe'])) {
    // Récupérer la nouvelle valeur de mot de passe
    $mot_de_passe = $_POST['mot_de_passe'];

    // Mettre à jour le mot de passe dans la base de données
    $query_update = "UPDATE personnes SET mot_de_passe = :mot_de_passe WHERE id_personne
= :user_id";
    $stmt_update = $db->prepare($query_update);
    $stmt_update->bindParam(':mot_de_passe', $mot_de_passe);
    $stmt_update->bindParam(':user_id', $personne);
    $stmt_update->execute();

    // Redirection vers la même page après la mise à jour
    header("Location: profil.php");
    exit();

}
```