

*Intégrer des bases de données
dans le web*



Ludovic Abraham

*Version 2 du 10.03.2024 16:28:00
Date de création du fichier 26.08.2022*

*Module du 01.02.2024 à
08.03.2024*



Table des matières

1	Analyse	1
1.1	Intro	1
1.1.1	Pourquoi ce projet ?	1
1.1.2	Fonctionnalités	1
1.1.2.1	Résistance aux injections	1
1.1.2.2	Visiteur	1
1.1.2.3	Utilisateur logué	1
1.2	Maquettes	1
1.2.1	Page d'accueil	1
1.2.2	S'inscrire	2
1.2.3	Se connecter	2
1.2.4	Dashboard	3
1.2.5	Ajouter un projet	3
1.3	Usecase	4
1.4	Diagramme d'activité	4
1.4.1	Login	4
1.4.2	Ajout de projet	4
1.5	Diagramme de séquence	4
1.5.1	Login	5
1.5.2	Ajout de projet	6
1.6	Diagramme entité-relation	6
2	Conception	8
2.1	Diagramme de classe partie cliente	8
2.2	Diagramme de classe partie serveur	8
2.3	Diagramme de séquence interaction	9
2.3.1	Login	10
2.3.2	Ajouter un projet	11
2.4	Pas de tests (injection SQL, HTML, JS)	11

2.5	Pas de test fonctionnels.....	12
2.6	Schéma relationnel de la base de données.....	13
3	Implémentation.....	14
3.1	Descente de code	14
3.1.1	Format de mes réponses en JSON	17
3.2	Tests fonctionnels	17
3.2.1	Injectons	17
3.2.2	Fonctionnalités	18
3.3	Problèmes rencontrés	19
3.3.1	Problème de CORS lors du login	19
3.3.2	Problème lors du lorsque le client doit contacter le php	20
3.3.3	Problèmes avec l'escape des caractères côté serveur	20
3.4	Hébergement sur EMFProd.....	20
3.4.1	Liens sur l'hébergement	21
4	Synthèse.....	22
4.1	Présentation réalisation	22
4.2	Différences entre planning et réalisation	22
4.3	Conclusion	22



Error! Use the Home tab to apply Titre 1 to the text that you want to appear here.

1 Analyse

1.1 Intro

Ce projet est un outil qui va permettre de faire de la gestion de projet, elle vise les devs principalement. Dans ce site on pourra créer des projets, et gérer les tâches qui y sont liés. Une fois la tâche créée on pourra éditer les détails, notamment ajouter les heures passé dessus.

1.1.1 Pourquoi ce projet ?

J'ai choisi de faire ce projet car il peut m'être très utile dans le cadre pro, et je compte également le faire évoluer d'avantage dans un futur proche.

1.1.2 Fonctionnalités

Voici la liste de fonctionnalités à mettre en place dans le cadre de ce projet :

1.1.2.1 Résistance aux injections

- Résistance aux injections HTML/CSS/JS côté client
- Résistance aux injections SQL côté serveur

1.1.2.2 Visiteur

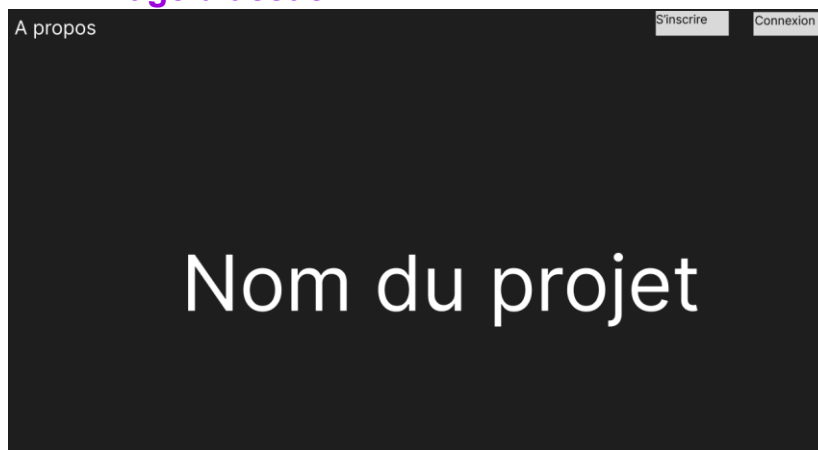
- Création d'un utilisateur
- Se connecter

1.1.2.3 Utilisateur logué

- Se déconnecter
- Ajouter un projet
- Ajouter une tâche
- Suppression d'une tâche
- Modification de l'état d'une tâche
- Édition du détail de la tâche (nom, description, temps passé dessus)

1.2 Maquettes

1.2.1 Page d'accueil



1.2.2 S'inscrire

Création de votre compte

Username

Mot de passe

Confirmer votre mot de passe

Déjà un compte ? connectez vous ici

S'inscrire

1.2.3 Se connecter

Connectez vous à votre compte

Username

Mot de passe

Pas de compte ? inscrivez-vous ici

Se connecter

1.2.4 Dashboard

Nom du projet ↓ Username

à faire	En cours	Terminé	Problèmes
+	+	+	+

1.2.5 Ajouter un projet

Ici c'est un peu particulier, il s'agit d'un forms qui apparaît au beau milieu du dashboard, avec un fond noir mais avec 60% de transparence pour faire comprendre au user qu'ils sont toujours dans le dashboard.

Nom du projet ↓ Ajouter un projet Username

à faire En cours Terminé Problèmes

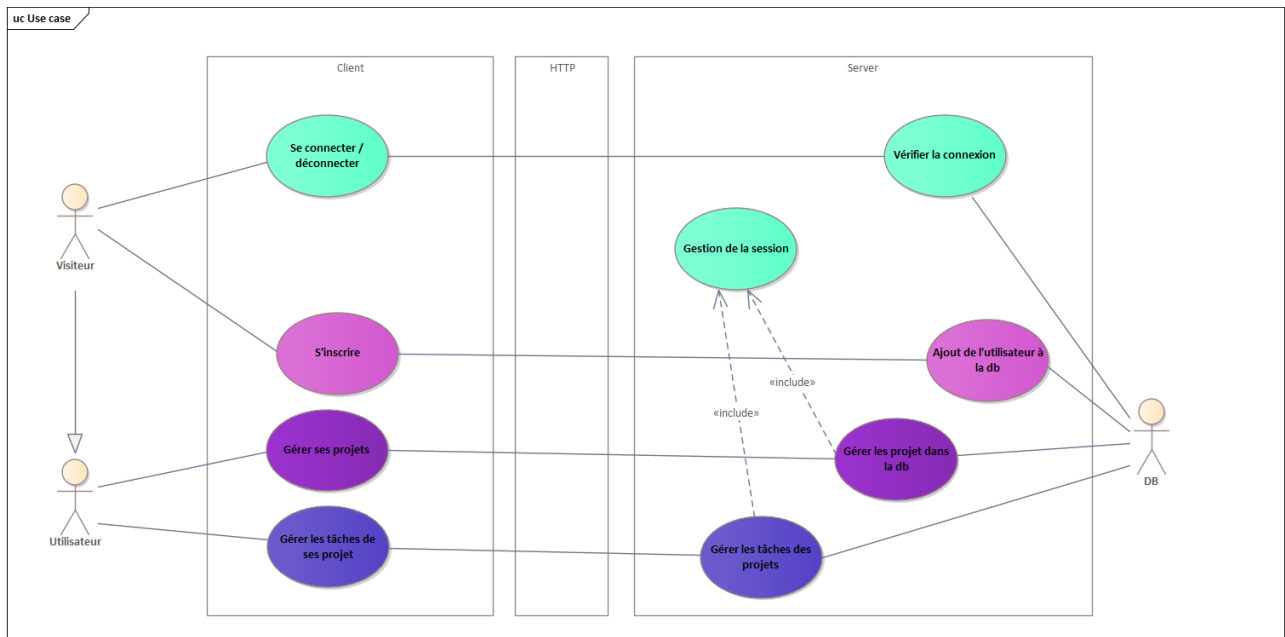
Création d'un projet

Nom du projet

Description du projet

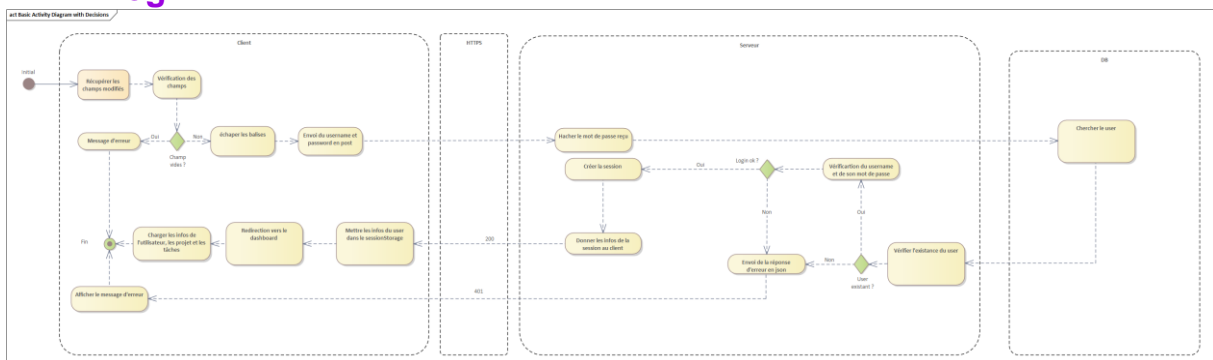
Ajouter le projet

1.3 Usecase

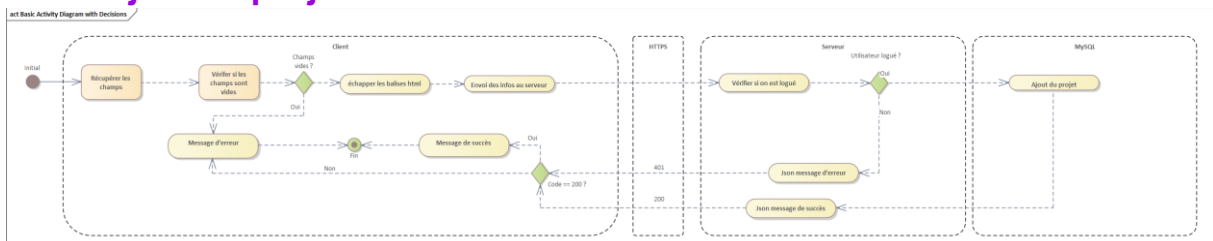


1.4 Diagramme d'activité

1.4.1 Login



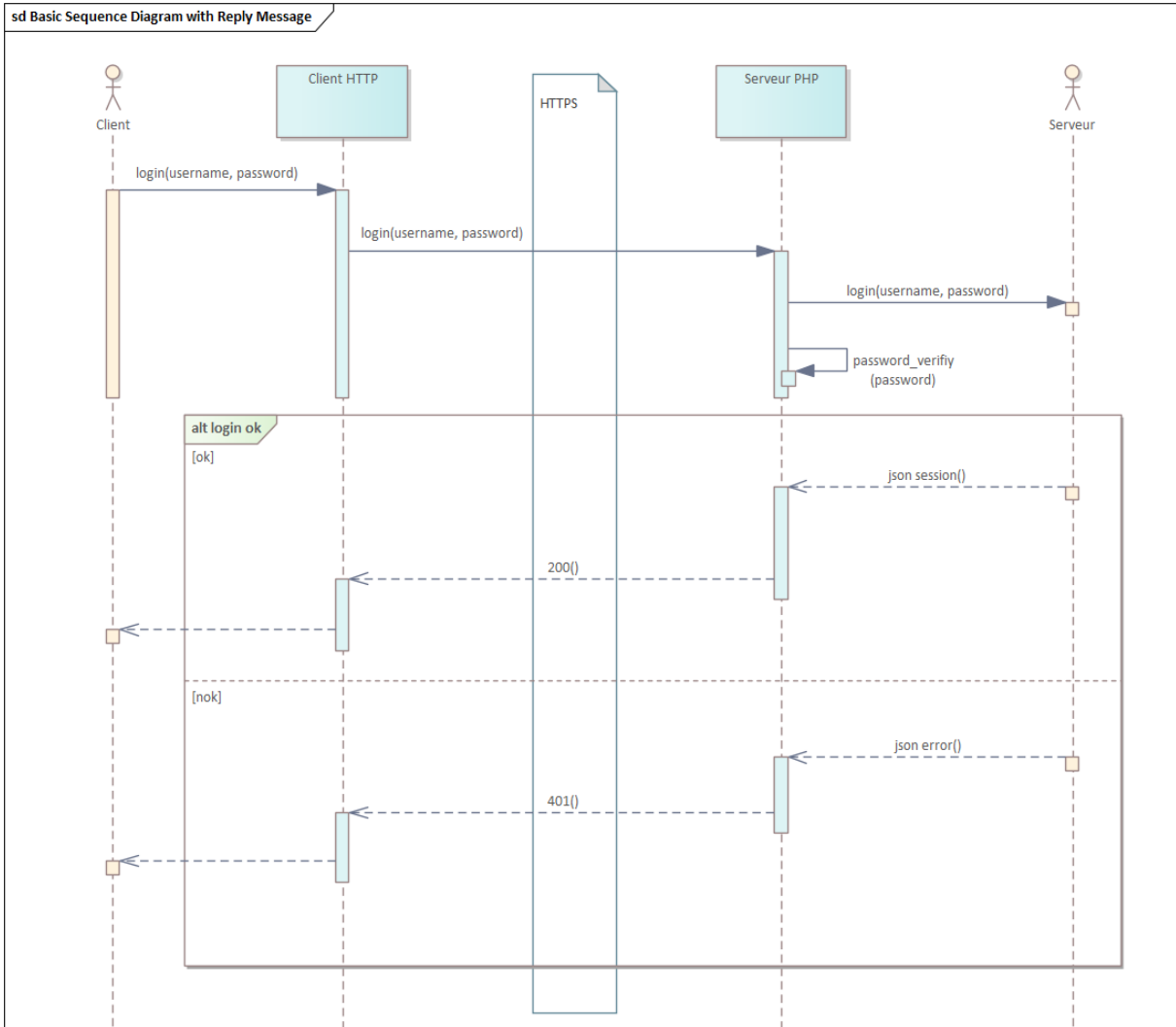
1.4.2 Ajout de projet



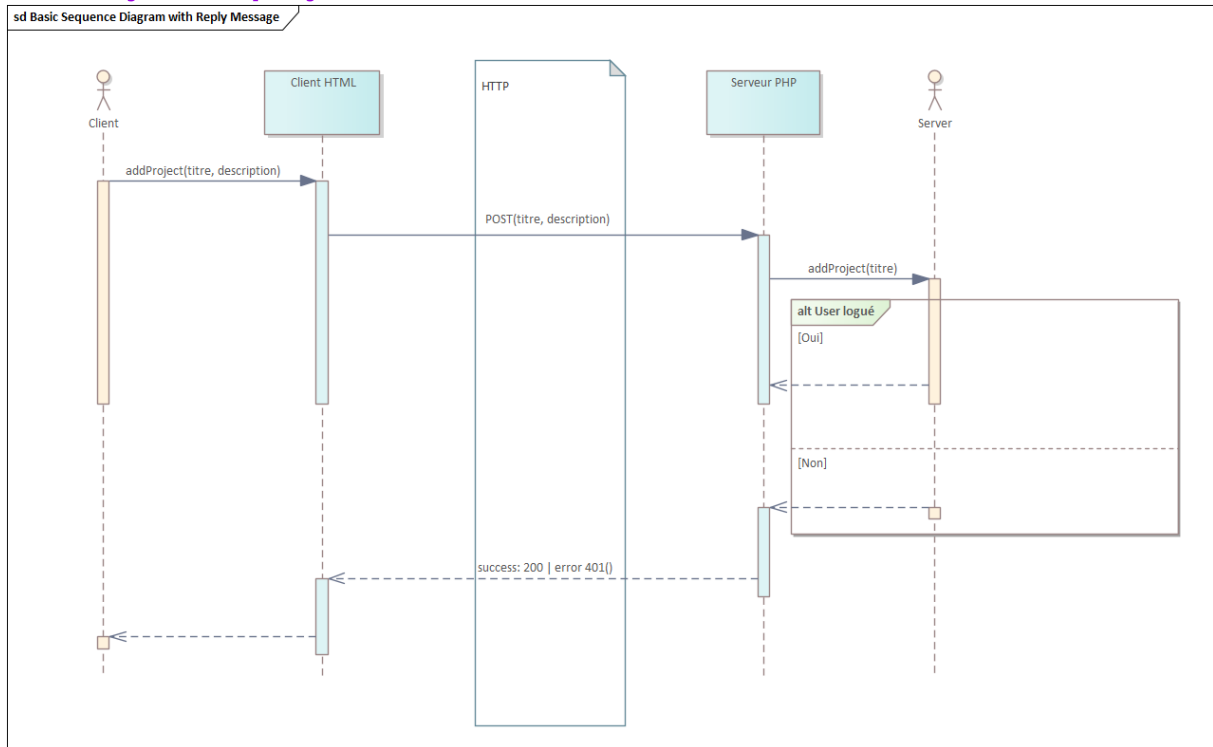
1.5 Diagramme de séquence

1.5.1 Login

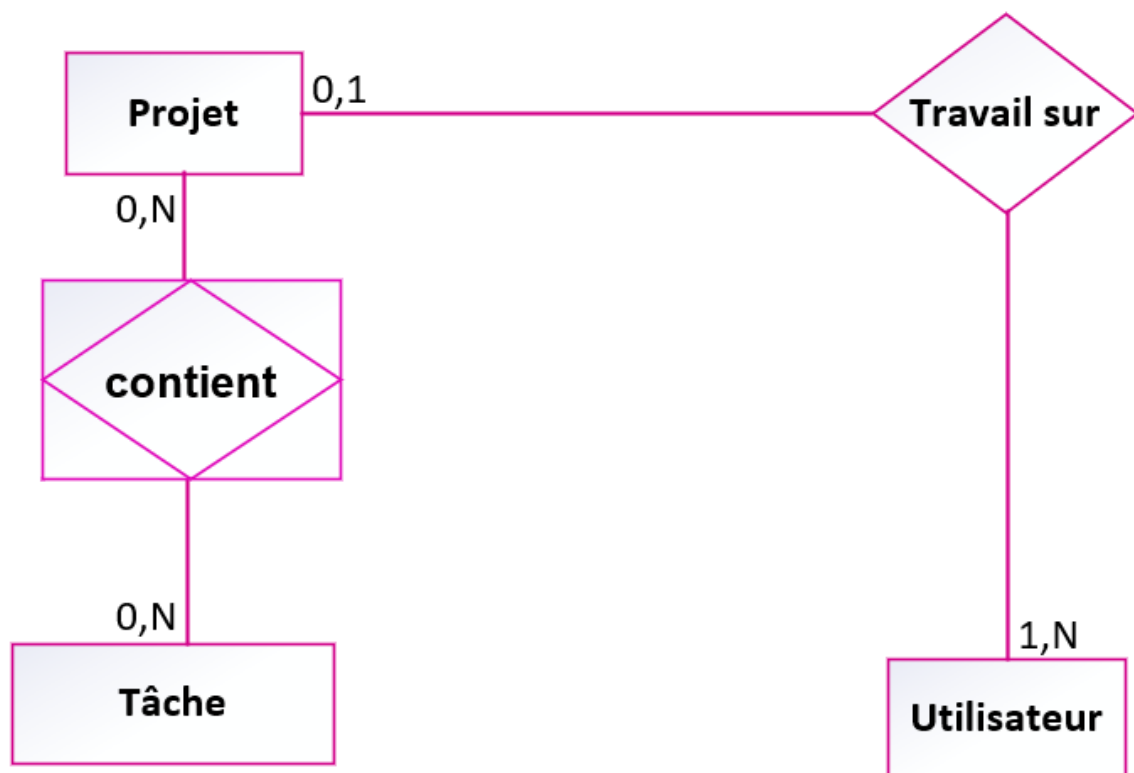
sd Basic Sequence Diagram with Reply Message



1.5.2 Ajout de projet



1.6 Diagramme entité-relation

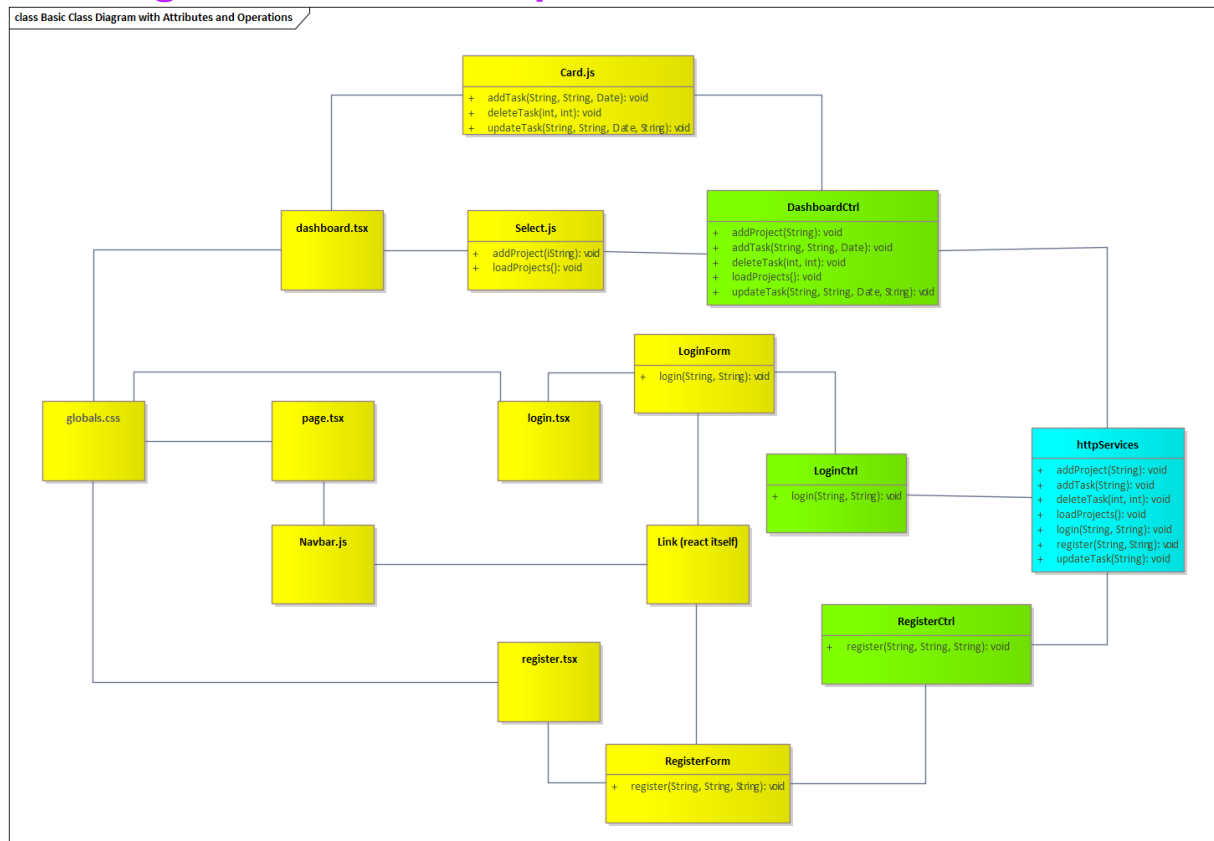


Correction : Une tâche peut être dans un seul et unique projet, et le projet peut avoir plusieurs tâches.

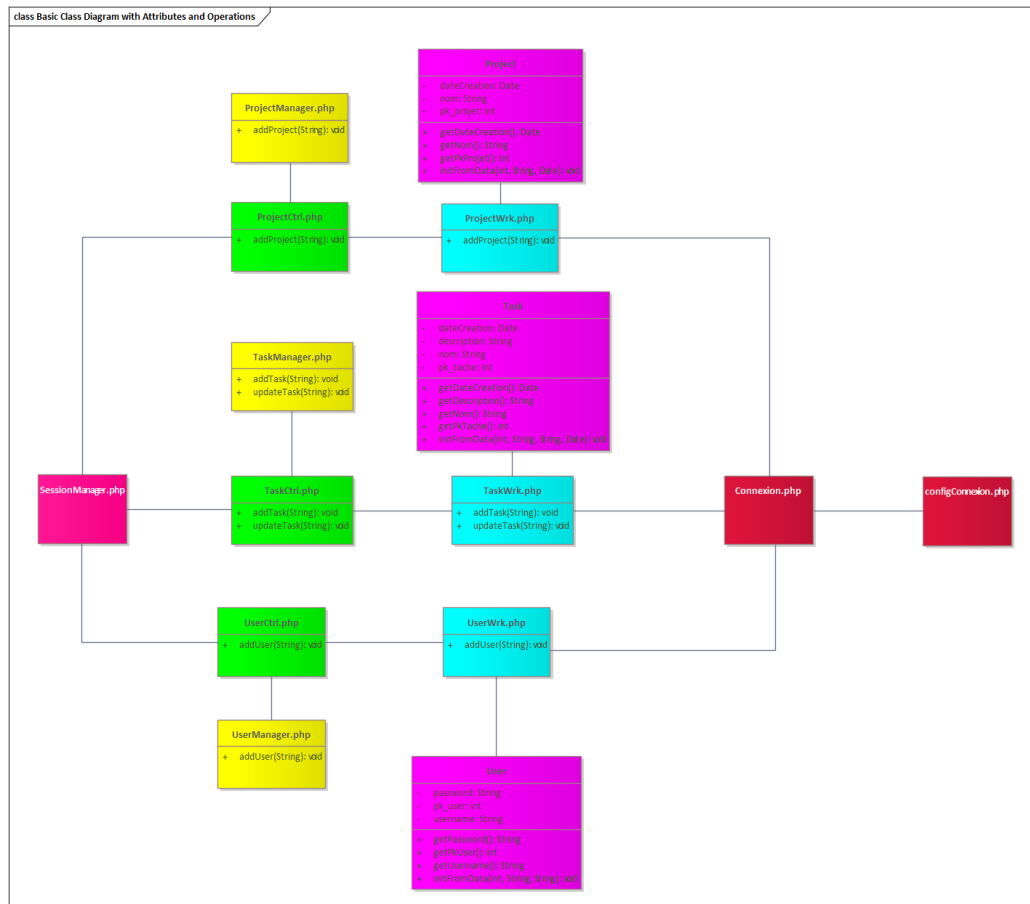
1.7 Planning

2 Conception

2.1 Diagramme de classe partie cliente

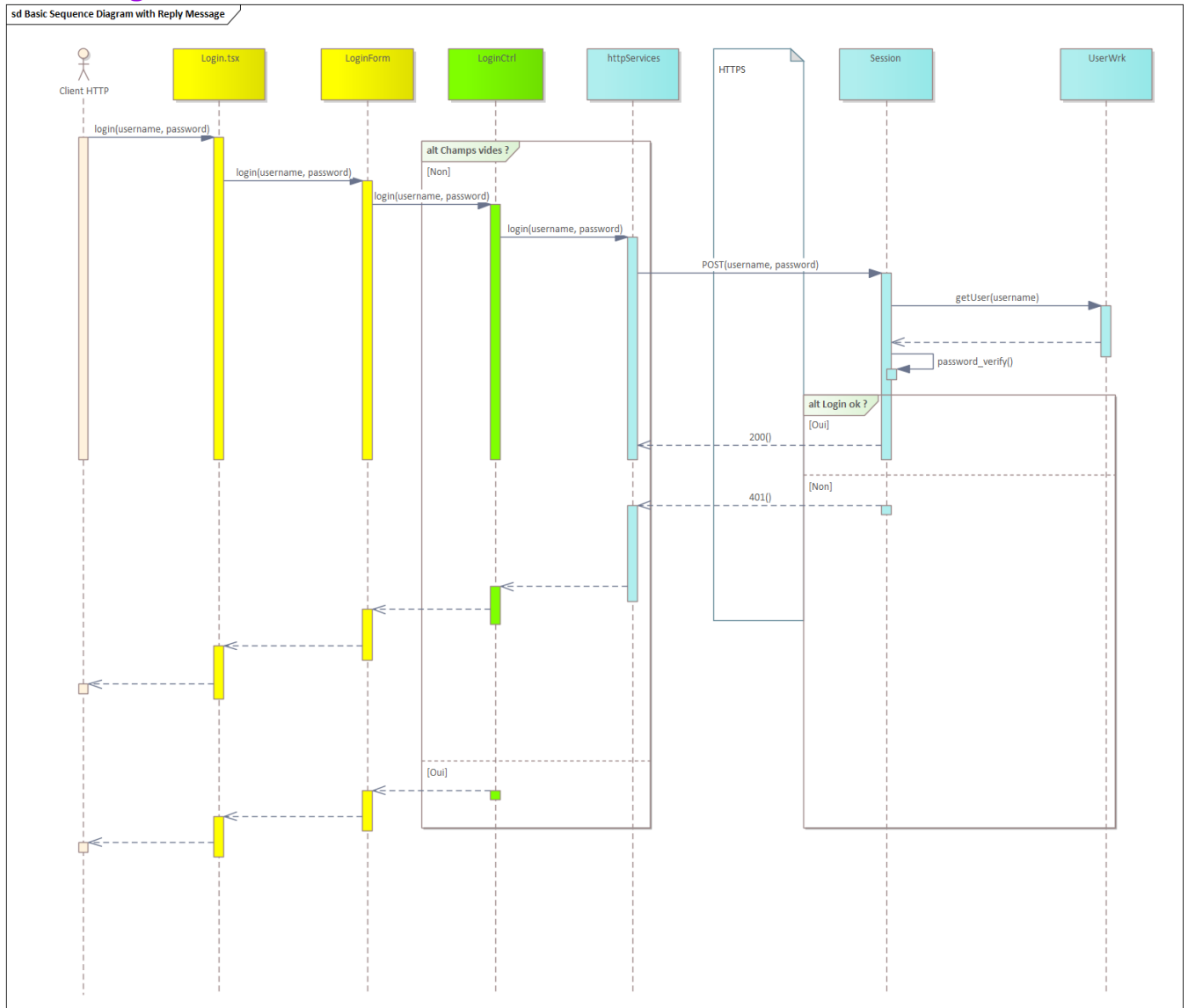


2.2 Diagramme de classe partie serveur

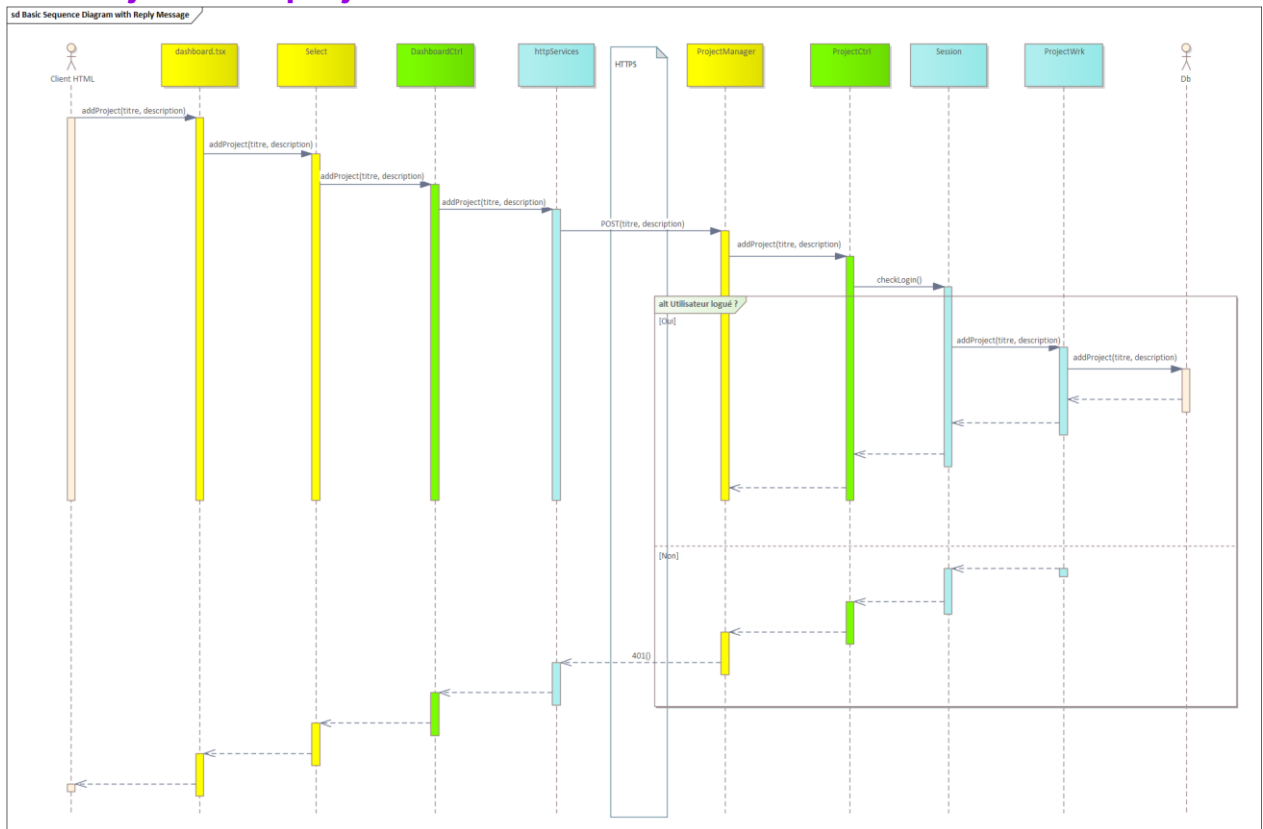


2.3 Diagramme de séquence interaction

2.3.1 Login



2.3.2 Ajouter un projet



2.4 Pas de tests (injection SQL, HTML, JS)

Voici les pas de tests à effectuer pour que le site soit le plus robuste possible :

Quel champs ?	Quel page ?	Résultat attendu	Résultat obtenu	Test validé ?
Username	Login	Pas d'injection possible		
Username	Inscription	Pas d'injection possible		
Mot de passe	Inscription	Pas d'injection possible		
Confirmer le mot de passe	Inscription	Pas d'injection possible		
Ajouter une tâche	Dashboard	Pas d'injection possible		
Modifier une tâche	Dashboard	Pas d'injection possible		
Ajouter un projet	Dashboard	Pas d'injection possible		

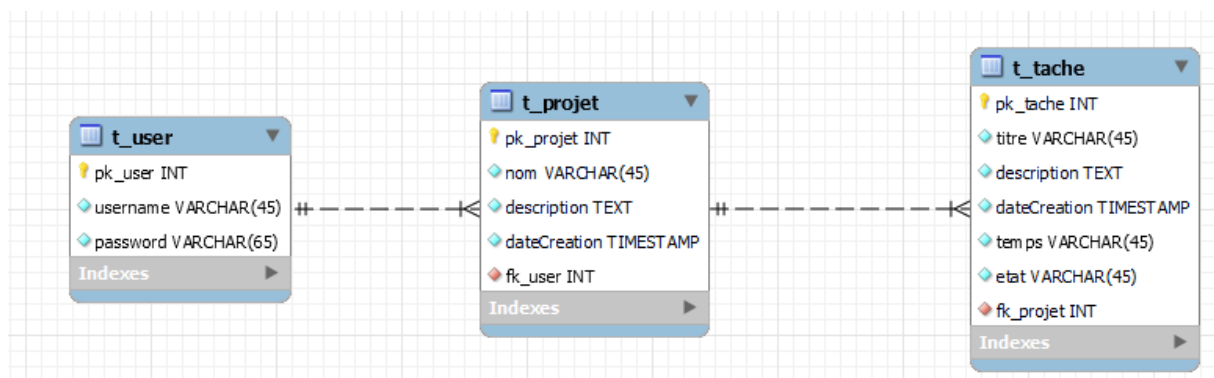
2.5 Pas de test fonctionnels

Voici les pas de tests pour ce qui concerne les fonctionnalités du projet :

Fonctionnalité	Libellé	Résultat attendu	Résultat obtenu	Test validé ?
Register	Avec un nom d'utilisateur disponible	Utilisateur créé		
Register	Avec un nom d'utilisateur déjà pris	Message d'erreur		
Register	Avec un mot de passe qui ne respecte pas les critères de complexité	Message d'erreur		
Register	Avec un mot de passe qui ne correspond pas à la confirmation du mot de passe	Message d'erreur		
Login	Avec un bon nom d'utilisateur et mot de passe	Login ok, redirection sur le dashboard		
Login	Avec un utilisateur qui n'existe pas	Message d'erreur vague		
Login	Avec un mauvais mot de passe mais un bon utilisateur	Message d'erreur vague		
Créer un projet	En étant connecté	Projet ajouté		
Créer un projet	Sans être connecté	Message d'erreur		
Créer une tâche	En étant connecté	Tâche ajoutée		
Créer une tâche	Sans être logué	Message d'erreur		

Déplacer une tâche	En étant logué	Tâche déplacé et modifié dans la db (Refresh pour s'assurer que ce soit bon)		
Déplacer une tâche	Sans être logué	Message d'erreur		
Se déconnecter	Avec un utilisateur logué	Déconnexion effectué et redirection dans la page d'accueil		
Accéder au dashboard	Sans être logué	Redirection vers l'accueil		

2.6 Schéma relationnel de la base de données



3 Implémentation

3.1 Descente de code

Pour cette descente de code j'ai choisi l'ajout d'un projet, comme pour les diagrammes de séquences et d'activités. Donc mon client envoie un POST avec les données en JSON. Ce que je vais devoir faire c'est réceptionner le json, le décoder et mettre les valeurs dans des variables.

```
case 'POST':

    $res = "";

    $name = initVariableFromJson("name");
    $description = initVariableFromJson("description");
    $fk_user = initVariableFromJson("fk_user");

    if (isset($name, $description, $fk_user)) {

        $res = $ctrl->addProject($name, $description, $fk_user);

    }

    echo $res;
    break;
```

Pour faire en sorte d'avoir les valeurs pour le \$name, \$description et \$fk_user j'ai utilisé cette méthode ci, elle se trouve dans tools/common.php :

```
function initVariableFromJson($key){

    $data = json_decode(file_get_contents("php://input"), true);

    return isset($data[$key]) ? $data[$key] : null;
}
```

Une fois qu'on a vérifié si les valeurs sont bel et bien là on passe la suite au contrôleur. Ce qu'il va faire c'est contrôler que l'on soit bien logué sur notre app et il va aussi construire les json de réponses en fonction de ce que le wrk va lui envoyer :

```
public function addProject($nom, $description, $fk_user)
{

    $res = '';

    $isLogged = SessionManager::getInstance()->checkServerLogin();

    if ($isLogged) {

        $wrk = new ProjectWrk();

        $success = $wrk->addProject($nom, $description, $fk_user);
```

```

        if ($success) {

            $res = writeJSONResponse(OK, "Projet ajouté: " . $nom,
array());

        } else {

            $res = writeJSONResponse(INTERNAL_SERVER_ERROR, "Un problème
est survenu lors de la création du projet", array());

        }

    } else {

        $res = writeJSONResponse(FORBIDDEN, "Utilisateur non logué",
array());

    }

    return $res;

}

```

Quand on est logué le contrôleur va passer la suite au worker. Le worker lui il va vérifier si le nom et la description s'il ont besoin d'un escape (Pour virer les balises html). Ensuite avec le strip_tags on les enlève. Une fois ça fait il va taper dans la db pour ajouter le projet. Puis on récupère le nombre de lignes affectés, si c'est différent de 1 c'est false et donc, l'ajout s'est mal passé. Le contrôleur il va construire les json en fonction de ce que le worker retourne.

```

public function addProject($name, $description, $fk_user)
{

    $nameNeedsEscapes = preg_match('/<[^>]+>/', $name);
    $descriptionNeedsEscapes = preg_match('/<[^>]+>/', $description);

    if ($nameNeedsEscapes <> 0)
        $name = strip_tags($name);
    if ($descriptionNeedsEscapes <> 0)
        $description = strip_tags($description);

    $params = array("name" => $name, "description" => $description,
"fk_user" => $fk_user);

    $affectedLines = Connexion::getInstance()->executeQuery("INSERT INTO
t_projet(nom, description, fk_user) VALUES (:name, :description, :fk_user);",
$params);

```

```

        return $affectedLines == 1;
    }

```

Quand le contrôleur il va construire ses json, il va utiliser cette méthode-ci :

```

function writeJSONResponse($code, $message, $data){

    http_response_code($code);

    // Définit L'en-tête Content-Type pour indiquer que La réponse est en JSON
    // avec L'encodage UTF-8
    header('Content-Type: application/json; charset=utf-8');

    // Encode Les données en JSON, en conservant Les caractères Unicode non
    // échappés
    $json = json_encode(array("code" => $code, "message" => $message, "body"
=> $data), JSON_UNESCAPED_UNICODE);

    // Gestion de L'erreur d'encodage JSON
    if ($json === false) {

        //Prendre des infos supplémentaires sur L'erreur d'encodage
        $jsonError = json_last_error_msg();

        // Génère une réponse d'erreur avec Le code 500 Internal Server Error
        // en cas d'échec d'encodage JSON
        $json = json_encode(
            array(
                "code" => INTERNAL_SERVER_ERROR,
                "message" => "Erreur lors de l'encodage en JSON : " .
                $jsonError,
                "body" => array()
            ), JSON_UNESCAPED_UNICODE);
    }

    // Retourne La réponse JSON générée
    return $json;
}

```

Quand on va écrire le json on va aussi aller taper dans les constantes pour récupérer les codes http que j'ai défini :

```

define("OK", 200);
define("BAD_REQUEST", 400);
define("UNAUTHORIZED", 401);
define("FORBIDDEN", 403);
define("NOT_FOUND", 404);
define("INTERNAL_SERVER_ERROR", 500);
define("NOT_IMPLEMENTED", 501);
define("SERVICE_UNAVAILABLE", 503);

```

3.1.1 Format de mes réponses en JSON








Mes réponses en JSON ont toujours un code (pour la réponse http envoyé), un message (De succès ou d'erreur) et un body (Pour les données que je veux que le client traite). Voici un exemple de retour:

```
{
  "code": 200,
  "message": "Connexion réussie",
  "body": {
    "pk_user": 13,
    "username": "KibiByte7"
  }
}
```

3.2 Tests fonctionnels

3.2.1 Injections







Voici les pas de tests à effectuer pour que le site soit le plus robuste possible :

Quel champs ?	Quel page ?	Résultat attendu	Résultat obtenu	Test validé ?
Username	Login	Pas d'injection possible	Pas d'injection possible	
Username	Inscription	Pas d'injection possible	Pas d'injection possible	
Mot de passe	Inscription	Pas d'injection possible	Pas d'injection possible	
Confirmer le mot de passe	Inscription	Pas d'injection possible	Pas d'injection possible	
Ajouter une tâche	Dashboard	Pas d'injection possible	Pas d'injection possible	
Modifier une tâche	Dashboard	Pas d'injection possible	Pas d'injection possible	
Ajouter un projet	Dashboard	Pas d'injection possible	Pas d'injection possible	

3.2.2 Fonctionnalités

Voici les tests des différentes fonctionnalités :

Fonctionnalité	Libellé	Résultat attendu	Résultat obtenu	Test validé ?
Register	Avec un nom d'utilisateur disponible	Utilisateur créé	Utilisateur créé	
Register	Avec un nom d'utilisateur déjà pris	Message d'erreur	Message d'erreur	
Register	Avec un mot de passe qui ne respecte pas les critères de complexité	Message d'erreur	Message d'erreur	
Register	Avec un mot de passe qui ne correspond pas à la confirmation du mot de passe	Message d'erreur	Message d'erreur	
Login	Avec un bon nom d'utilisateur et mot de passe	Login ok, redirection sur le dashboard	Login ok, redirection sur le dashboard	
Login	Avec un utilisateur qui n'existe pas	Message d'erreur vague	Message d'erreur vague	
Login	Avec un mauvais mot de passe mais un bon utilisateur	Message d'erreur vague	Message d'erreur vague	
Créer un projet	En étant connecté	Projet ajouté	Projet ajouté	
Créer un projet	Sans être connecté	Message d'erreur	Message d'erreur	

Créer une tâche	En étant connecté	Tâche ajouté	Tâche ajouté	
Créer une tâche	Sans être logué	Message d'erreur	Message d'erreur	
Déplacer une tâche	En étant logué	Tâche déplacé et modifié dans la db (Refresh pour s'assurer que ce soit bon)	Tâche déplacé et modifié dans la db (Refresh pour s'assurer que ce soit bon)	
Déplacer une tâche	Sans être logué	Message d'erreur	Message d'erreur	
Se déconnecter	Avec un utilisateur logué	Déconnexion effectué et redirection dans la page d'accueil	Déconnexion effectué et redirection dans la page d'accueil	
Accéder au dashboard	Sans être logué	Redirection vers l'accueil	Redirection vers l'accueil	

3.3 Problèmes rencontrés

3.3.1 Problème de CORS lors du login

J'ai une des problèmes de cors quand mon client essayait de contacter le serveur. En regardant sur oneNote, j'ai vite compris qu'il me fallait l'équivalent de xhrFields with credentials en réact. D coup je me suis référé dans la doc de axios. Et j'ai ajouté la ligne en surbrillance pour régler le problème:

```

async login(username, password) {
  try {
    const res = await axios.post(baseUrl + 'session.php', {
      "action": "login",
      "username": username,
      "password": password
    },
    {
      headers: {
        'Content-Type': 'application/json'
      },
      withCredentials: true
    });
  }
}

```

```

    return res.data;
  } catch (error) {
    return error.response.data;
  }
},

```

3.3.2 Problème lors du lorsque le client doit contacter le php

J'ai eu ce problèmes lorsque je voulais faire mes premières communications entre mon client et mon serveur, il se trouvait que les requêtes OPTIONS sont bloqués par le CORS, en demandant à Kevin il m'a dit que c'était une vérification faite sur le client. J'ai donc été voir les headers à mettre sur le serveur et j'ai mis ces lignes pour régler le problème :

```

header('Access-Control-Allow-Origin: http://localhost:3000');
header("Access-Control-Allow-Methods: POST, GET, OPTIONS");
header("Access-Control-Allow-Headers: Content-Type");
header('Access-Control-Allow-Credentials: true');

```

3.3.3 Problèmes avec l'escape des caractères côté serveur

J'ai eu ce problème lorsque que je voulais enlever les balises du username de mon utilisateur, en fait quand il n'y a pas de balises strip_tags me retourne une erreur, j'ai du faire un regex pour savoir si le nom de l'utilisateur avec besoin d'un escape :

```

$usernameNeedsEscapes = preg_match('/<[>]+>/', $username);

if ($usernameNeedsEscapes <> 0)
    $username = strip_tags($username);

```

3.4 Hébergement sur EMFProd

L'hébergement a posé problème car je peux pas mettre le projet réact directement dans le serveur et faire en sorte qu'il le compile pour moi. J'ai du donc modifier le paramétrage de mon application comme ceci :

```

/** @type {import('next').NextConfig} */
const nextConfig = {
  reactStrictMode: false,
  output: "export",
  basePath: "/151/client",
  trailingSlash: true
};

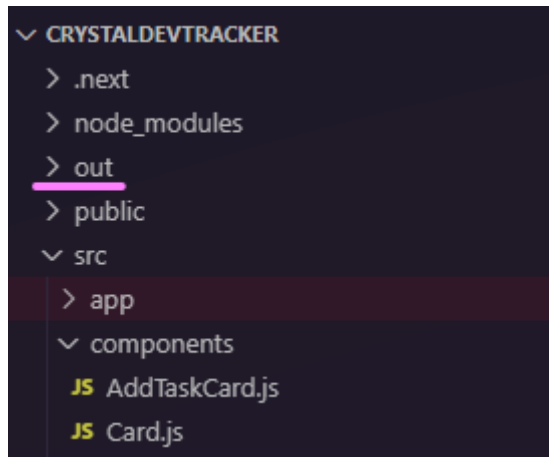
export default nextConfig;

```

Ensuite j'ai tapé :

```
npm run build
```

Ça aura pour effet de construire le projet et de placer la version « statique » directement dans le répertoire out (équivalent au dist quand on démarre un projet Java).



Et c'est le contenu de tout ce répertoire que je vais devoir placer dans /public_html/151/client/

Pour ce qu'il en est des liens les voici :

3.4.1 Liens sur l'hébergement

- [Page d'accueil](#)
- [Page d'inscription](#)
- [Page de connexion](#)
- [Page du tableau de bord](#) (Accessible seulement si l'utilisateur est connecté)
- [Page d'aide](#)

je les ai également posés dans le readme de mon github

Error! Use the Home tab to apply Titre 1 to the text that you want to appear here.

4 Synthèse

4.1 Présentation réalisation

Il y a eu pas mal de différences entre la conception et la réalisation surtout avec le client, car je devais pouvoir charger les projets et les tâches avec le composant en question au départ, j'ai du centraliser mes instances de contrôleur pour pouvoir faire passer plus loin les infos. J'ai réussi à intégrer toutes les fonctionnalités sauf la modification du détail de la tâche par manque de temps.

4.2 Différences entre planning et réalisation

Ce qui m'a pris le plus de temps que prévu c'est la conception, j'ai encore du mal à ne pas partir dans tout les sens quand il s'agit d'analyser les besoin et comment fonctionne certaines fonctionnalités. La partie code elle, elle aura pris moins de temps que prévu.

4.3 Conclusion

J'ai beaucoup aimé apprendre le réact et consolider mes connaissances en php entre EMFStages et ce projet-ci j'ai remarqué qu'il y a eu une nette progression en terme de code. Malgré les difficultés que j'ai rencontrés j'ai tout de même réussi à faire un projet qui soit potable, même s'il y a quelques améliorations que je peux faire dans un futur proche.