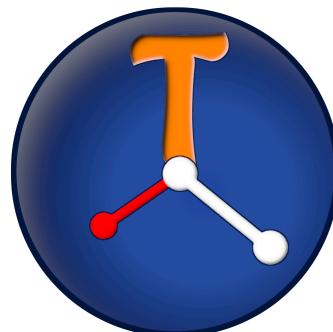


Eric CORBISIER

Juin 2024

Concepteur Développeur d'Application

Tranquillo Organizer
App



Résumé

Dans ce dossier sont présentées les différentes étapes de la conception jusqu'au développement de l'application ainsi que la préparation du poste de travail, des outils adaptés au besoin de l'application.

L'application **Tranquillo Organizer** est spécialement conçue pour répondre aux besoins des personnes ayant un Trouble Déficitaire de l'Attention avec Hyperactivité (TDAH) et des personnes avec un Trouble du Spectre Autistique (TSA). Néanmoins, elle est ouverte à toute personne ayant des besoins particuliers dans la gestion de son quotidien.

Elle a pour mission d'offrir une approche innovante de la planification, favorisant une expérience personnalisée et adaptive.

Les utilisateurs pourront créer des rappels flexibles, visualiser les délais, et ajuster les paramètres en fonction de leurs besoins spécifiques, avec des options comme le mode nuit et un journal de suivi.

Abstract

This document presents the various stages from design to development of the application, as well as the preparation of the workstation and tools tailored to the application's needs.

The **Tranquillo Organizer** is specially designed to meet the needs of individuals with Attention Deficit Hyperactivity Disorder (ADHD) and Autism Spectrum Disorder (ASD). However, it is also open to anyone with specific needs in managing their daily life.

Its mission is to offer an innovative approach to planning, providing a personalized and adaptive experience. Users will be able to create flexible reminders, visualize deadlines, and adjust settings according to their specific needs, with options like night mode and a tracking journal.

TABLE DES MATIÈRES

Projet	8
Compétence mise en oeuvre	9
Expression du besoin	10
1.1. Les users stories	10
1.2. Les personas	10
Présentation de l'entreprise	11
➤ L'idée	11
➤ La naissance	11
Gestion de projet	11
➤ Planning et suivi	11
★ Le WorkFlow	11
★ Les Issues	12
★ Le tableau Kanban	12
★ Definish Of Done :	12
L'environnement humain	12
★ Auto-évaluation des Compétences	12
★ Définition des Rôles et Responsabilités	13
★ Planification Personnelle	13
★ Auto-motivation	13
★ Engagement	13
➤ Qualité du Processus :	13
★ Conformité aux Standards	13
★ Gestion des Risques	13
★ Amélioration Continue	13
Les spécifications Fonctionnelles	14
Les Contraintes	15
1. Les Problématiques	15
2. Contraintes	15
3. Objectifs	15
L'architecture logicielle du projet	15
1. Les couches logiques	15
2. Les 3 couches logicielles	16
la Conception	18
Choix de l'Interface Utilisateur	19
1. NativeScript	19
2. Svelte Native	19
Utilités	19
Le Zoning	20
Les Wireframes	21
Vue sur une tablette Vue sur un mobile	21
Le Visuel	22
➤ LA CHARTE GRAPHIQUE	22
1. Le logo	22
2. Les couleurs	22
3. La typographie	22
4. Les styles graphiques	22

La Maquette Statique	23
➤ LE RENDU FINAL	23
(cf Visuel réel : annexe C.2)	23
la Couche Métier	24
Le Choix des Langages	25
1. Mes choix	25
1.1. PHP	25
1.2. Symfony	25
La sécurité avant tout	26
➤ Exemples	26
la Couche Données	27
Rappel	28
Identification des Besoins Utilisateur :	28
Identification des Besoins de l'Application :	28
Conception de la Base de Données	28
➤ MERISE : Une Méthode d'Analyse et de Conception	28
Analyse et Conception des Données :	28
Les Trois Niveaux de MERISE :	29
❖ Inconvénients	29
la Modélisation Conceptuelle des Données	29
Le Choix du Système de Gestion de Bases de Données (SGBD) :	29
La Normalisation des Données :	30
La Migrations et Scripts de Déploiement :	30
La Sécurité et Sauvegardes :	30
Modèle Conceptuel des Données (MCD)	31
1. Les Use Case	31
2. Le Modèle Conceptuel de Données (MCD)	31
3. Les types de base de données	33
Le modèle Logique de Données	34
Cardinalité des Associations :	35
Le modèle Physique de Données	36
➤ La Dénormalisation et le clé secondaire	36
➤ Cas Concret : MPD pour MariaDB en SQL pour Tranquillo	37
le Code (Réalisation)	38
Les diagrammes	38
Structure de l'API (Couche métier)	39
ORM (Object-Relational Mapping)	41
Les Classes	42
Mise en place de la Sécurité	43
1. Validation des données	43
2. Métier (API)	43
3. Sécurisation des données	43
les Tests	45
Les différents tests	45
Le plan des tests	46
Les tests Unitaires	47
Exemples d'un test manuel sur Thunder	47
le Déploiement	48
1. NativeScript	49
2. PHP Symfony	49
3. MariaDB avec Docker	50

Veille Sécurité	51
ATTAKUES CSRF	52
➤ API REST	52
➤ CSRF	52
→ Qu'est ce que c'est ?	52
→ Connexion à l'API REST sans JWT	53
→ Connexion à l'API REST avec JWT	53
→ Problème de sécurité côté client	55
→ Quelle peut être la solution ?	56
➤ Pour résumer	57
Pour conclure	58
Résumé du Projet Tranquillo Organizer	58
ANNEXES	59
A. Projet	59
A.1. User storie	59
A.2. Personas	60
B. Kanban	61
B.1. Liste issues (workflow)	61
B.2. Exemple Issues (ou tickets)	61
B.3. le tableau Kanban	62
B.4. Cadre de travail SCRUM	63
B.5. Exemple de bonne pratiques	63
C. La conception	64
C.1. Svelte Native	64
C.1.1. Structure	64
C.1.2. Code en Svelte	65
C.1.3. Utilisation du token JWT	66
C.1.4. Code avec Typescript	66
C.2. Visuel réel sur mobile Android	68
C.3. Couche métier	69
C.3.1. Architecture	69
C.3.2. Code	69
C.3.2.1. JWT	69
C.3.2.2. Sécurité	70
C.3.2.2.1. Injection SQL :	70
C.3.2.2.2. Cross-Site Scripting (XSS) :	71
C.3.2.2.3. Cross-Site Request Forgery (CSRF) :	71
C.3.2.2.4. Brute Force	71
C.3.2.2.5. Man-in-the-Middle (MITM)	71
C.3.2.2.6. Dénie de Service (DoS) et Dénie de Service Distribué (DDoS)	71
C.3.2.2.7. Vol de Session	71
C.3.2.3. Service	72
C.3.2.4. DTO	72
C.3.2.5. Controller	73
C.3.2.6. Entity	73
C.4. Base de Données	74
C.4.1. Code SQL	74
C.4.2. Compte utilisateur MariaDB pour l'API	74
C.4.3. Fixtures	75

C.4.3.1. Base	75
C.4.3.2. Utilisation	75
C.4.4. Users	76
C.4.4.1. Architecture	76
C.4.4.2. Valeur	76
C.4.5. Tasks	77
C.4.5.1. Architecture	77
C.4.5.2. Valeur	77
C.4.5.3. Relationnel	77
D. Les Diagrammes	78
D.1. MCD	78
D.2. MLD	78
D.3. Classe	79
D.4. Use Case	80
E. Les Tests	80
E.1. Interface Utilisateurs	80
E.2. Couche métier	81
E.3. Tests Unitaire	82
E.3.1. Le Code	82
E.3.2. Résultats	82
F. Le Déploiements	83
F.1. Déploiement Interface Utilisateur	83
F.2. Déploiement Serveur Métier	83
F.3. Déploiement Base de données	84
Définitions	85
• Qu'est-ce qu'un n-uplet ?	85
• Qu'est-ce que la Scalabilité	85
1. Scalabilité Verticale (ou Scaling Up)	85
2. Scalabilité Horizontale (ou Scaling Out)	86
➤ Le fichier SQL	86
➤ JWT	86
Ressources	87
To Conclude	88
Summary of the Tranquillo Organizer Project	88
Remerciements	90

Le Projet



Compétence mise en oeuvre

N° Fiche AT	Activités types	N° Fiche CP	Compétences professionnelles	
1	Développer une application sécurisée	1	Installer et configurer son environnement de travail en fonction du projet	<input checked="" type="checkbox"/>
		2	Développer des interfaces utilisateur	<input checked="" type="checkbox"/>
		3	Développer des composants métier	<input checked="" type="checkbox"/>
		4	Contribuer à la gestion d'un projet informatique	<input checked="" type="checkbox"/>
2	Concevoir et développer une application sécurisée organisée en couches	5	Analyser les besoins et maquetter une application	<input checked="" type="checkbox"/>
		6	Définir l'architecture logicielle d'une application	<input checked="" type="checkbox"/>
		7	Concevoir et mettre en place une base de données relationnelle	<input checked="" type="checkbox"/>
		8	Développer des composants d'accès aux données SQL et NoSQL	<input checked="" type="checkbox"/>
3	Préparer le déploiement d'une application sécurisée	9	Préparer et exécuter les plans de tests d'une application	<input checked="" type="checkbox"/>
		10	Préparer et documenter le déploiement d'une application	<input checked="" type="checkbox"/>
		11	Contribuer à la mise en production dans une démarche DevOps	<input checked="" type="checkbox"/>

Les besoins, définis en fonction des problématiques des utilisateurs, varient notamment en présence de handicaps, ce qui a conduit à utiliser des user stories et des personas pour adapter l'application, accessible en français et en anglais, aux situations de chaque personnel..

1.1. Les users stories

Les user stories sont des descriptions simples et courtes des fonctionnalités du point de vue de l'utilisateur. Elles suivent généralement le format :

- En tant que [type d'utilisateur]
- je veux [objectif]
- afin de [raison/bénéfice]

En voici deux appliquées à mon projet :

★ **User Story 1:**

- **En tant que** personne atteinte de TDAH,
- **Je veux** créer des rappels flexibles avec des options de répétition personnalisées,
- **Afin de** m'assurer que je n'oublie pas les tâches importantes même pendant les jours les plus chargés.



* cf Autres users story : annexe A.1

★ **User Story 2:**

- **En tant qu'** utilisateur souffrant du Syndrome d'Asperger,
- **Je veux** pouvoir visualiser clairement le temps restant jusqu'à une échéance,
- **Afin de** mieux gérer mon temps et réduire mon stress.

Ces users stories m'aident à définir les fonctionnalités clés de l'application et à prioriser les développements en fonction des besoins des utilisateurs.

Les users stories sont une étape cruciale car ils permettent de créer des personas.

1.2. Les personas

Les personas, dérivés du terme latin signifiant "masque de théâtre", représentent des archétypes d'utilisateurs dotés d'un nom, d'un visage et décrits avec précision en termes de besoins, d'objectifs et de tâches.

Ils permettent de construire une vision claire des utilisateurs et facilitent l'analyse des besoins individuels, guidant ainsi la conception en identifiant les caractéristiques et attentes des utilisateurs cibles, ce qui est précieux pour préparer les fonctionnalités du projet.

J'ai choisi deux personas avec des besoins et spécificités très variés. Ils ont été adaptés à partir d'éléments d'une enquête effectuée auprès de mes proches et de mes collègues.

1.2.1. Marc

atteints d'un trouble

MARC *

Problèmes
développeur informatique talentueux travaillant dans une entreprise de technologie. Marc a un TSA. Il peut souvent rencontrer des difficultés dans la gestion du temps et des tâches.

Défis
"J'aime quand tout est bien organisé et structuré. Cela me permet de me concentrer sur ce que je fais et d'être plus productif dans mon travail de développement."

Age
entre 25 et 30 ans

Profession
Développeur informatique

1.2.2. Sophie

travaillant avec des enfants

SOPHIE*

Problèmes
enseignante passionnée qui travaille avec des enfants dont certains sont atteints de TDAH (Trouble Déficitaire de l'Attention avec ou sans Hyperactivité), elle jongle avec de nombreuses tâches au quotidien, y compris la préparation des cours, la gestion de la classe et le suivi des progrès des élèves. Malgré son emploi du temps chargé, elle essaie de maintenir un équilibre entre son travail et sa vie personnelle.

Age
entre 40 et 45 ans

Profession
Professeur des écoles

* cf Personas complet : annexe A.2



> L'idée

💡 Étant moi-même touché par la difficulté à m'organiser au quotidien sans rien oublié, j'ai recherché une application qui réponde à mes besoins.

🧐 Jusqu'à présent, il me faut utiliser deux, voire trois applications différentes pour parvenir à m'accompagner dans mon quotidien.

📝 Après avoir fait une enquête auprès de mon entourage, le projet a tout de suite généré un effet positif avec des propositions concrètes de ce que l'on peut attendre d'une telle application.

> La naissance

⌚ Tout commence à mes 15 ans lorsque je reçois une formation informatique. Ma première création fut de développer un annuaire téléphonique sur le système MS-Dos.



⌚ J'ai poursuivi mon expérience en développant des jeux sur ma calculatrice.

⌚ Le Bac Pro m'a permis d'apprendre toute l'architecture d'une programmation réussie, en passant par les attentes clients, à l'offset puis à la programmation du robot.

⌚ Je me suis également passionné pour le VBA d'office Excel, créant des tableurs dynamiques.

⌚ Puis je me suis tourné vers le web. J'ai créé mon premier site docalarmes.fr (partage de documentations de centrales d'alarmes et vidéosurveillance pour mes collègues), puis corbisier.fr (pour partager des photos avec la famille et les amis).

⌚ Ma dernière conception est le site professionnel pour l'activité de mon épouse : <https://lescorbycats.fr>



⌚ Tout cela m'a amené à suivre la formation CDA et ouvert la perspective de concevoir et développer Tranquillo Organizer .

Gestion de projet

> Planning et suivi

Pour mener à terme mon projet, dont l'achèvement est prévu d'ici deux ou trois ans, j'ai choisi de le gérer avec les outils suivants :

★ Le WorkFlow



Le workflow structure les étapes nécessaires à l'achèvement d'une tâche ou d'un projet, permettant de visualiser clairement le processus, d'éviter les oubliés, de progresser constamment, et de diviser les projets complexes en tâches gérables pour un travail plus organisé et moins accablant.

Pour cela, j'utilise GitHub, qui comprend l'outil **Issues** pour créer des tickets, ainsi que l'outil **Project**, permettant notamment de créer un tableau Kanban.

cf Liste WorkFlow : annexe B.1

★ Les Issues

Les issues sont des éléments critiques pour gérer les tâches, les bugs, ou toute autre activité nécessitant une attention particulière.

Même lorsqu'on travaille seul, les issues permettent de consigner toutes les idées, problèmes ou tâches à traiter.

Elles me servent de rappel et m'aident à prioriser les travaux en cours.

De plus, elles fournissent une trace écrite des progrès et des décisions, facilitant ainsi la réflexion sur le processus et l'amélioration continue.

Nous en venons à l'utilisation d'un tableau Kanban.

FRONT Ajout du module d'inscription #78

Closed I #79 ecorbierSimplon/tranquillo Public

ecorbierSimplon opened 2 weeks ago edited by ecorbierSi

Création du module d'inscription.
Ce module contiendra :

Nom du label	Nom technique
Nom	lastname
Prenom	firstname
Email	email
Mot de passe	password
Confirmation du mot de passe	password_repeat

cf Exemples d'Issues : annexe B.2

★ Le tableau Kanban

Un tableau Kanban, outil visuel performant du cadre SCRUM (cf annexe B.4), organise et suit les tâches en offrant une vue d'ensemble claire des tâches 'à faire', 'en cours', 'en test' et 'terminées', aidant ainsi à gérer les priorités, éviter le multitâche excessif et mesurer les progrès.

En plaçant la première partie de mon WorkFlow dans la colonne BackLog et en utilisant des sprints d'une semaine, je maintiens une continuité du projet et respecte les délais et objectifs, tout en identifiant les goulets d'étranglement en déplaçant les tickets d'une colonne à l'autre.

The screenshots illustrate the use of a Kanban board for project management. The first screenshot shows the backlog section with several tasks listed. The second screenshot shows a task moved to the 'In progress' section, highlighted with a red circle. The third screenshot shows the completed section with several tasks listed.

L'utilisation d'un workflow, des issues et d'un tableau Kanban m'ont grandement permis d'améliorer l'organisation, la productivité et la clarté du travail à effectuer.

★ Define Of Done :

Une fois que la tâche décrite sur le ticket correspond au Define Of Done, la fonctionnalité peut être déployée en production et le ticket mis dans la colonne Done.

L'environnement humain

Travaillant seul, je suis dans une dynamique d'auto-gestion.

★ Auto-évaluation des Compétences

Je dois identifier mes propres compétences et évaluer où je pourrais avoir besoin de renforcement ou de formation supplémentaire, en particulier dans les technologies que j'utilise comme PHP/Symfony et NativeScript.



★ Définition des Rôles et Responsabilités

Je dois définir clairement mes propres rôles et responsabilités dans le projet. Cela inclut le développement de la logique applicative, la conception de l'interface utilisateur, les tests, la gestion du projet, etc.

★ Planification Personnelle

J'ai aussi établi et fait vivre un plan de projet détaillé qui définit les étapes à suivre, les délais à respecter et les objectifs à atteindre. Cela m'aidera à rester organisé et à maintenir ma motivation.

★ Auto-motivation

Pour rester motivé et engagé dans mon projet, je célèbre les petites victoires, m'auto-récompense pour les objectifs atteints et cherche des communautés en ligne pour partager mes progrès et obtenir des retours,

★ Engagement

Mon engagement est également présente en utilisant des solutions respectueuses de l'environnement pour limiter mon impact écologique, comme par exemple :

Hébergement Vert : En choisissant un fournisseur d'hébergement web qui utilise des sources d'énergie renouvelable ou qui compense son empreinte carbone.

Optimisation des ressources : En optimisant l'utilisation des ressources informatiques pour réduire la consommation d'énergie, comme la consolidation des serveurs, la virtualisation et l'utilisation de conteneurs.



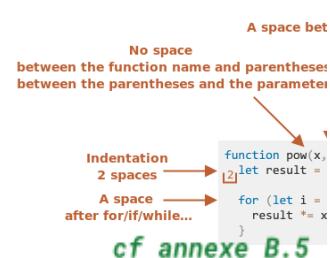
Éco-conception Logicielle : je souhaite concevoir des logiciels éco-responsables, efficaces en énergie, optimisés pour la mémoire et performants, en minimisant les requêtes superflues et en maîtrisant la gestion des bases de données.

➤ Qualité du Processus :

★ Conformité aux Standards

Nous suivons les standards de développement et les bonnes pratiques

- les conventions de codage, les revues de code, les tests automatisés



★ Gestion des Risques



- Identifier et gérer les risques tout au long du projet pour minimiser les impacts négatifs potentiels, en incluant la gestion de la sécurité contre les vulnérabilités comme les attaques XSS, CSRF et les injections SQL.
 - la mise en place de mesures de sécurité avancées telles que l'authentification par jeton et le hachage des mots de passe, garantissant ainsi la confidentialité et l'intégrité des données sensibles de l'application.

★ Amélioration Continue

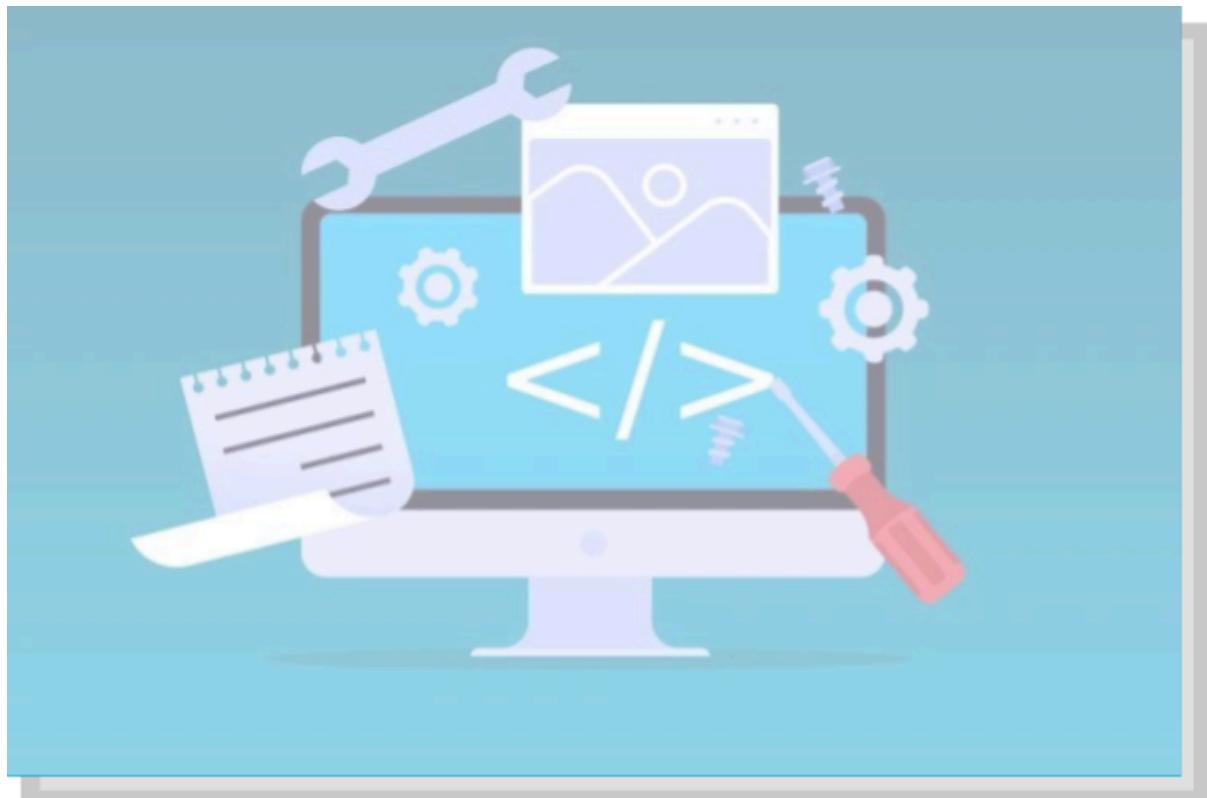
Mettre en place des processus d'amélioration continue pour identifier les points à améliorer et mettre en œuvre des actions correctives.



Cela nécessite également de rester constamment à jour sur les dernières avancées en matière de sécurité, ainsi que sur la connaissance approfondie des éléments constitutifs de mon architecture logicielle.

Les Spécifications

Fonctionnelles



1. Les Problématiques

Les personnes avec un Trouble Déficitaire de l'Attention avec Hyperactivité (TDAH), un Trouble du Spectre Autistique (TSA), d'autres troubles de mémorisation, ou d'organisation quotidienne, peuvent rencontrer des difficultés à structurer leur quotidien, se concentrer, respecter les délais, et nécessitent des rappels fréquents et des interfaces simples pour gérer leur temps sans stress.

2. Contraintes

- 2.1. La plus importante c'est moi même. Il me faut garder un fil rouge et rester sur cette ligne.
- 2.2. Une autre contrainte est le temps imparti pour réaliser mon projet, prioriser les objectifs de manière optimale.
- 2.3. J'ai choisi l'application mobile pour son accessibilité constante, ses notifications en temps réel, sa personnalisation selon les besoins individuels, et son utilisation des fonctionnalités natives des smartphones.
- 2.4. Puis, étant sur une démarche bénévole et d'un projet open-source, j'ai dû favoriser des outils gratuits ou dont le coût est gérable dans mon budget personnel.

3. Objectifs

Mon projet vise à développer une application mobile de gestion du quotidien, intégrant des rendez-vous et tâches depuis une interface simple et intuitive, avec des outils de rappel personnalisables et des fonctionnalités comme le mode nuit et un journal de suivi pour améliorer l'expérience utilisateur.

L'architecture logicielle du projet

1. Les couches logiques

J'ai utilisé une architecture séparée en 3 couches logiques :

1. la couche de présentation (ou interface utilisateur),
2. la couche métier (ou logique applicative)
3. et la couche de données (ou persistance).

Cette organisation rend le code plus modulaire et assure une nette séparation des responsabilités.

- **Scalabilité (cf chapitre Définitions)**

Chaque couche peut être adaptée indépendamment des autres, permettant ainsi de répondre de manière plus efficace à des besoins croissants en performances et en charge.

- **Réutilisation du Code**

Les fonctionnalités métier peuvent être utilisées à plusieurs reprises dans diverses parties de l'application ou dans d'autres projets, ce qui réduit le temps de développement et améliore la cohérence du système.

- **Facilité de Test**

Chaque couche peut être testée de façon indépendante, simplifiant ainsi la mise en place de tests unitaires et de tests d'intégration. Cette approche permet de détecter plus facilement les erreurs et les anomalies, assurant ainsi une meilleure qualité du logiciel.

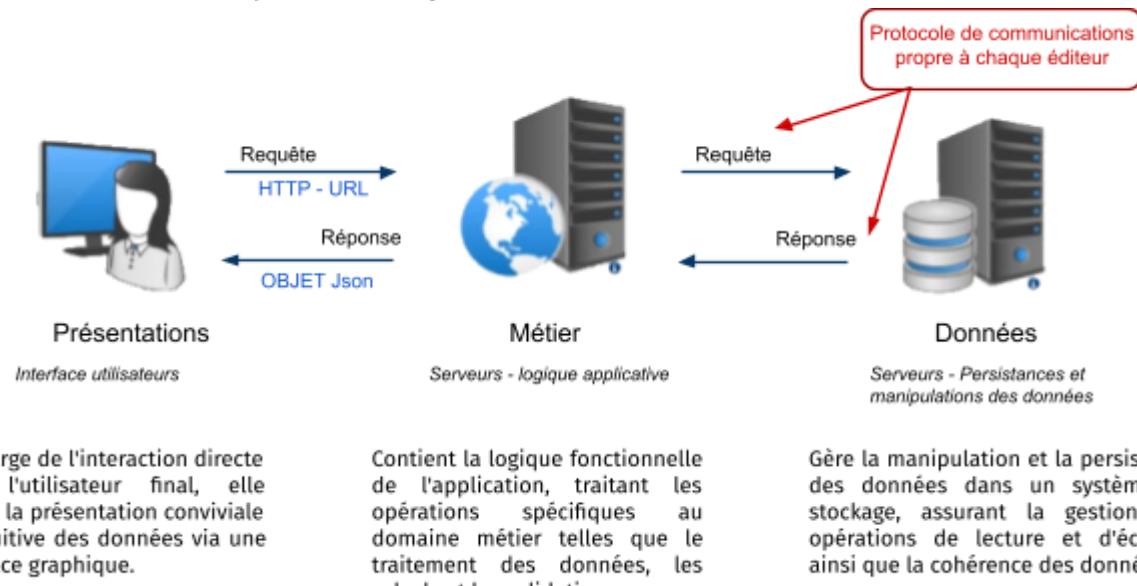
- **Flexibilité Technologique**

Cette architecture en 3 couches offre une grande souplesse dans le choix des technologies utilisées pour chaque couche. Par exemple, différentes technologies peuvent être employées pour la couche de présentation (telles que HTML/CSS/JavaScript ou des frameworks comme Angular ou React), la couche métier (telles que PHP/Symfony) et la couche de données (telles que MySQL ou MongoDB), en fonction des besoins et des contraintes spécifiques du projet.

De plus, elle permet le déploiement d'une couche sans nécessiter le redémarrage de l'ensemble des couches.

2. Les 3 couches logicielles

Voici les éléments de chaque couche logicielle :

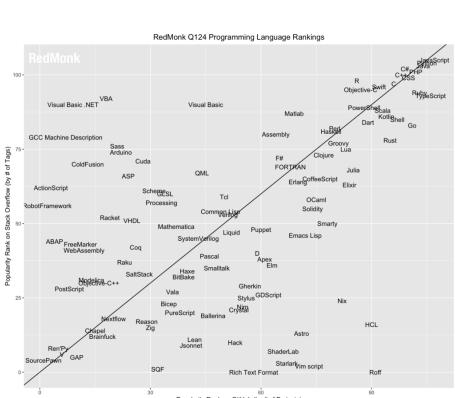


2.1. Choix des langages de programmation

Le choix du langage de programmation pour notre application repose sur plusieurs critères clés. L'un des principaux est que l'application doit être accessible via un mobile ou une tablette.

2.1.1. Comparaison des Technologies Web

Étant donné la multitude de technologies web disponibles, nous avons limité notre comparaison aux plus populaires et largement utilisées, en nous basant sur les statistiques de **RedMonk** du premier trimestre 2024. Ce classement reflète les grandes tendances parmi les développeurs.



1. JavaScript
2. Python
3. Java
4. PHP
5. C#
6. TypeScript

La liste des 20 langages de programmation les plus utilisés est restée assez stable, avec peu de changements parmi les premières places, ce qui montre un environnement résistant aux nouvelles tendances, même de l'IA.

- **Avantages des Langages Populaires**

Opter pour des langages très utilisés présente plusieurs avantages significatifs :

→ Robustesse et Fiabilité

En utilisant les conseils et les bonnes pratiques de développeurs expérimentés, on peut créer des applications plus robustes et fiables.

→ Rapidité de Développement

La disponibilité de nombreux outils, bibliothèques et frameworks permet de coder plus rapidement et efficacement.

→ Facilité de Maintenance

La maintenance et l'évolution de l'application sont simplifiées, car il est plus facile de trouver des développeurs compétents et des solutions aux problèmes courants.

→ Support et Documentation

Un langage populaire bénéficie d'un large support communautaire et de ressources documentaires, facilitant le développement et la résolution de problèmes.

2.1.2. Choix d'utiliser un framework

Un framework est une structure préétablie pour le développement de logiciels qui inclut des bibliothèques, des outils et des bonnes pratiques, facilitant et accélérant le développement en fournissant des solutions prêtes à l'emploi pour des tâches courantes et en imposant une organisation cohérente pour un code structuré et maintenable.



★ Avantages de l'utilisation d'un framework

L'utilisation d'un framework standardise et structure le développement, accélérant la création d'applications avec des outils intégrés, améliorant la sécurité et bénéficiant d'un large support communautaire.

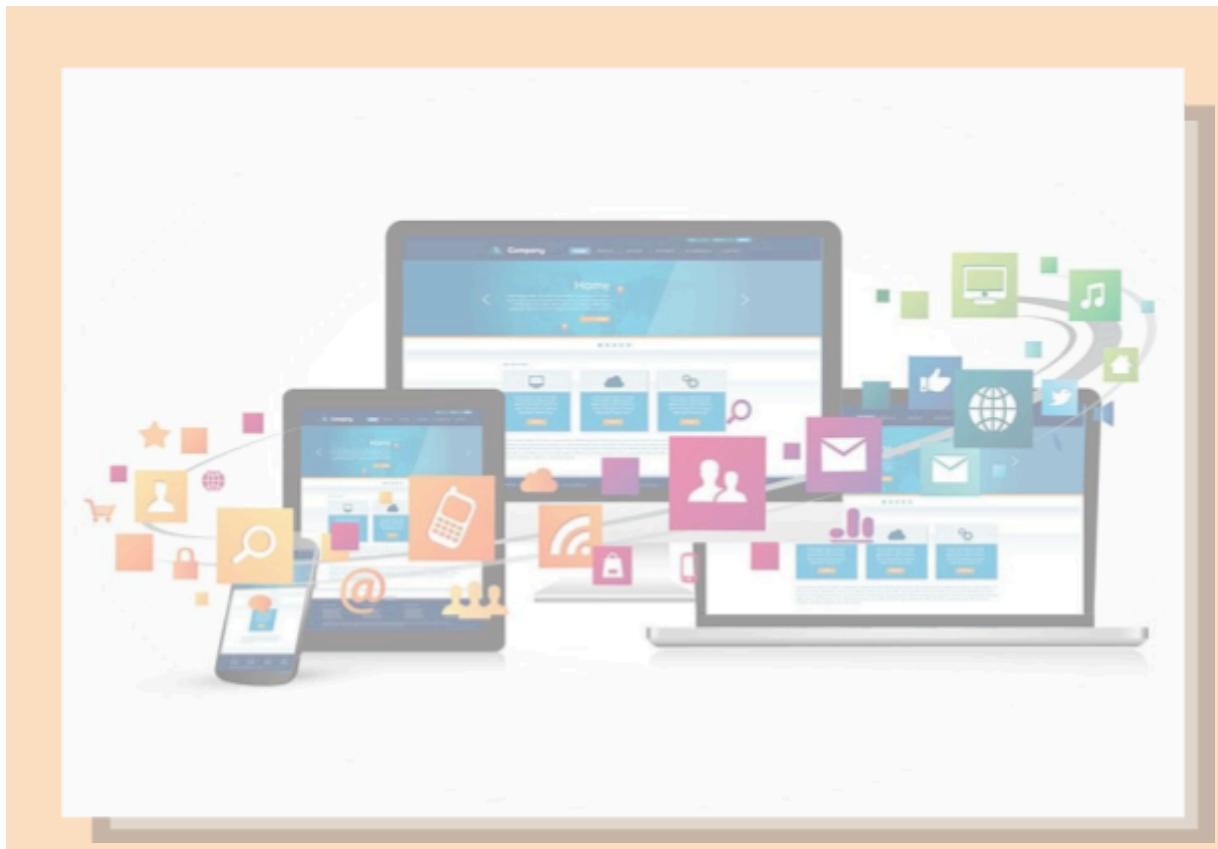
❖ Inconvénients de l'utilisation d'un framework

Les frameworks peuvent être difficiles à apprendre pour les débutants, introduire de la lourdeur inutile, limiter la flexibilité avec leurs conventions, et nécessiter des ajustements significatifs lors des mises à jour.

Mon choix d'un framework est une décision stratégique basée sur les besoins spécifiques de l'application, les fonctionnalités requises, l'infrastructure existante. En évaluant ces critères, je peut choisir le framework le plus adapté pour garantir le succès et la maintenance à long terme de l'application

1) la

Conception



Choix de l'Interface Utilisateur

J'utilise une interface mobile compatible Android et iPhone avec les frameworks :



1. NativeScript

NativeScript est un framework open-source permettant de développer des applications mobiles natives pour iOS et Android avec JavaScript, TypeScript, ou Angular, et dans ce projet, nous utilisons TypeScript avec le framework Svelte.



2. Svelte Native

Svelte Native est une extension de NativeScript qui utilise le framework Svelte pour créer des applications mobiles. Svelte compile le code à la construction, produisant des applications légères et performantes.

★ Pourquoi Svelte Native

Svelte Native offre d'excellentes performances avec un code JavaScript optimisé, une syntaxe claire, une gestion de l'état réactive intégrée, et surtout sans couche intermédiaire, ce qui rend les applications rapides et réactives.

❖ Inconvénients de Svelte Native



Cependant, Svelte Native a un écosystème moins développé avec moins de bibliothèques, de composants, de support communautaire, de plugins pour les fonctionnalités natives et une documentation peu abondante.

NativeScript et Svelte Native permettent de créer des applications mobiles performantes, réactives et maintenables, en maximisant la réutilisation du code et en réduisant les coûts de développement.

(cf *Code et architecture : annexe C.1*)

Utilités

Je souhaite concevoir l'application en intégrant plusieurs éléments essentiels pour offrir une expérience utilisateur optimale :

L'ergonomie et la navigation

Une navigation intuitive est primordiale. Les menus, les liens et les boutons doivent être bien organisés et faciles à trouver. L'ergonomie du site doit permettre aux utilisateurs de trouver rapidement ce qu'ils cherchent.

La mise en page et la structure

Une mise en page équilibrée et cohérente est importante. J'utilise des grilles pour organiser le contenu et veille à ce que les éléments visuels (textes, images, vidéos) soient disposés de manière logique.

La lisibilité et la typographie

Je choisis des polices lisibles et adapte leur taille pour une lecture confortable, en évitant les couleurs de texte trop vives sur des arrière-plans contrastants.

La compatibilité mobile

Je m'assure que mon site est responsive, c'est-à-dire qu'il s'adapte correctement aux différents appareils (ordinateurs, tablettes, smartphones).

La cohérence visuelle

Je respecte une palette de couleurs, un style graphique et une identité visuelle cohérente sur toutes les pages du site.

L'accessibilité

Je m'assure que mon site est accessible à tous, y compris aux personnes handicapées, en respectant les normes d'accessibilité WCAG.

La sécurité

Je protège les données des utilisateurs en utilisant des protocoles HTTPS et en mettant à jour régulièrement mon site.

Pour répondre au mieux à ses critères, j'ai utilisé

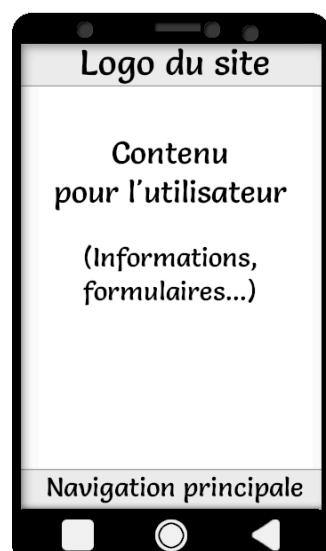
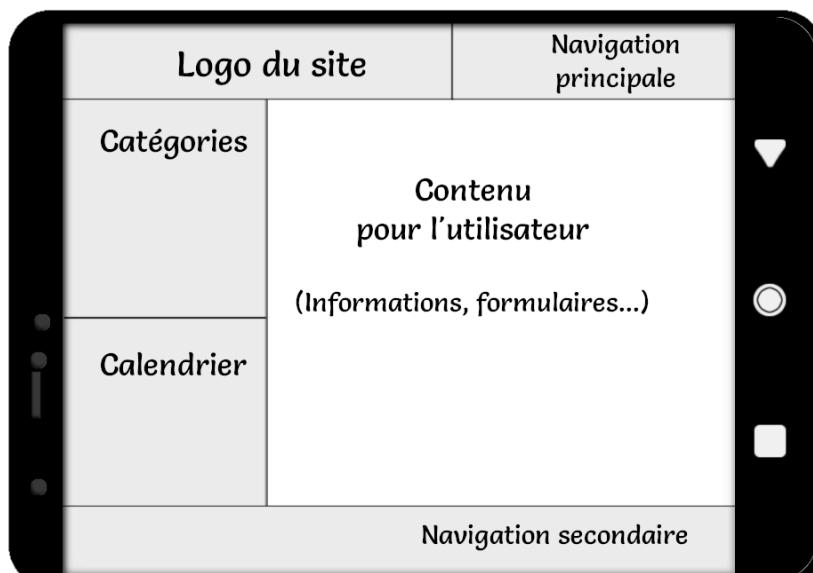
- le [zoning](#),
- le [wireframe](#),
- une [charte graphique](#)
- une [maquette statique](#).

Le Zoning

Le zoning dans la conception d'un site web organise la structure et l'agencement des différentes sections, améliorant ainsi l'expérience utilisateur.

Cette étape préliminaire optimise l'espace disponible et prépare le terrain pour des designs visuels cohérents et efficaces.

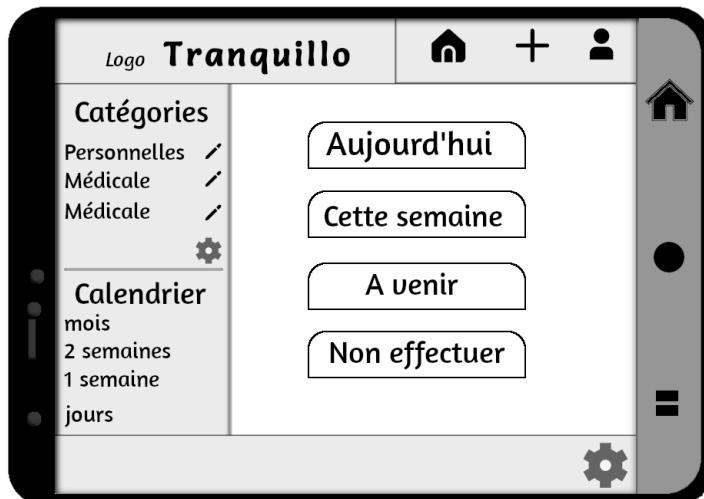
Établissons la base du site :



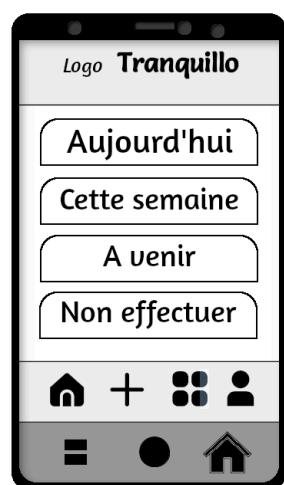
Les wireframes dans la conception d'un site web sont des schémas simples qui définissent la structure et la disposition des éléments sur chaque page, facilitant la visualisation du flux de navigation et l'interaction utilisateur.

Ils servent de guide pour les développeurs et les designers, assurant une mise en page fonctionnelle avant d'ajouter des éléments visuels détaillés.

Vue sur une tablette



Vue sur un mobile



Dans la représentation de Tranquillo ci-dessus, nous observons le changement d'état entre la page d'accueil et la page de création d'une tâche.

Nous pouvons observer également le visuel sur des appareils de tailles différentes comme un mobile et une tablette.

➤ LA CHARTE GRAPHIQUE

Je veille à ce que la charte graphique assure la cohérence visuelle de tous les supports de communication de la marque, en définissant les éléments visuels comme le logo, les couleurs, les typographies et les styles graphiques.

Cela facilite le travail des designers et des équipes marketing, garantissant une communication uniforme et professionnelle sur différents supports.

1. Le logo

- ❖ *L'emblème*
- ❖ *Le logo Typographique*



Tranquillo Organizer
App

2. Les couleurs

Pour rester cohérent, les couleurs reprennent celles choisies pour le logo.



3. La typographie

Le choix c'est porté sur des polices d'écritures sans serif et faciles à lire et reposante pour les yeux :

- | | | |
|------|------------|------------|
| 3.1. | le logo | : Salsa |
| 3.2. | Les titres | : Amaranth |
| 3.3. | Le texte | : Fira |

4. Les styles graphiques

4.1. Le css (*scss dans mon application*)

Il est convenu d'utiliser la librairie **Tailwind** qui a été adaptée pour fonctionner avec le code pour application mobile (qui diffère du code html).

De plus, il est responsive.

4.2. Les icônes

J'utilise une librairie connue, complète et gratuite : **Font Awesome 5**

	fas fa-bell
	far fa-bell
	fas fa-bell-slash
	far fa-bell-slash

	fas fa-calendar
	far fa-calendar
	fas fa-calendar-alt
	far fa-calendar-alt

	fas fa-fingerprint
	fas fa-id-badge
	far fa-id-badge
	fas fa-id-card

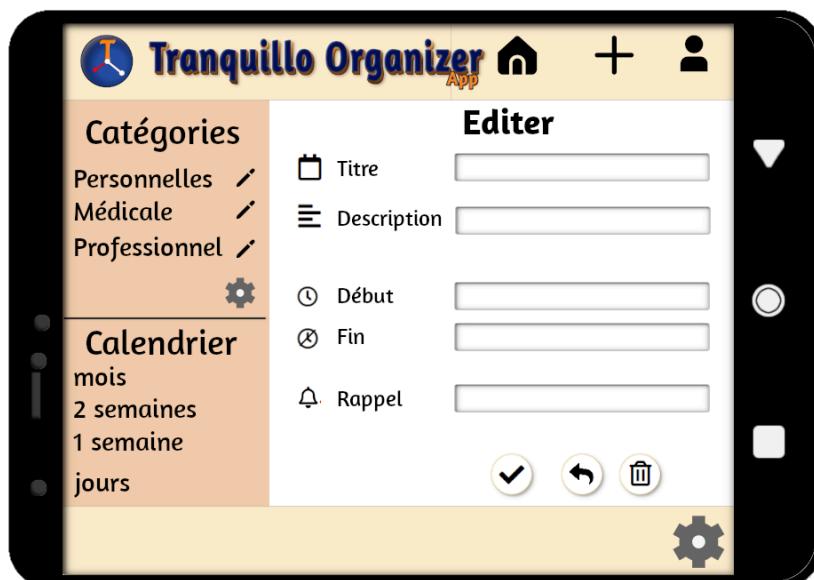
➤ LE RENDU FINAL

La maquette statique permet de visualiser et de structurer le design d'un site web ou d'une application avant le développement, facilitant la détection de problèmes potentiels.

Elle sert de support de communication entre les concepteurs et les développeurs, assurant que tous partagent la même vision (entre moi et moi-même).

Mais surtout, en permettant des ajustements précoce basés sur les retours d'utilisateurs et des parties prenantes, elle économise du temps de développement et des coûts.

Vue sur une tablette



Vue sur un mobile



(cf Visuel réel : annexe C.2)

2) la

Couche Métier



Pour la partie métier, j'ai choisi de développer une API Rest (voir le chapitre Veille Sécurité).

1. Mes choix

J'ai choisi d'utiliser le langage PHP et le framework Symfony pour ce projet. Disposant déjà d'un serveur et d'un site web fonctionnant en PHP, j'ai vu en cette opportunité une occasion idéale pour approfondir mes connaissances en Symfony. En intégrant ce framework, je vise à tirer parti de ses fonctionnalités avancées.

1.1. PHP

★ Avantages de PHP

PHP est populaire et bien supporté, facile à apprendre, idéal pour le développement web, avec une vaste communauté, de nombreuses ressources, et des mises à jour régulières pour améliorer performance et sécurité.

❖ Inconvénients de PHP

Cependant, PHP est critiqué pour des problèmes de sécurité liés à de mauvaises pratiques de codage, des erreurs de débogage dues à sa nature dynamique, une syntaxe incohérente, un manque de fonctionnalités avancées, et une performance moindre pour les applications à grande échelle.

J'ai choisi (cf 1. Mes Choix) PHP pour, entre autres, les raisons suivantes:

- Simplicité et Accessibilité
 - Large Communauté
 - Déploiement simple

PHP fonctionne avec la plupart des serveurs web et systèmes d'exploitation, et intègre facilement diverses bases de données.

1.2. *Symfony*

★ Avantages de Symfony

Symfony est un framework PHP robuste et flexible avec une architecture modulaire, une grande communauté, une vaste documentation, et des composants réutilisables pour créer des applications web performantes et sécurisées.

❖ Inconvénients de Symfony

Cependant, Symfony peut être difficile à apprendre pour les débutants, avoir des problèmes de performance sans optimisation, nécessiter plus de temps de développement à cause de sa modularité, et les mises à jour peuvent exiger des ajustements significatifs du code.

J'ai choisi (cf 1. Mes Choix) Symfony pour, entre autres, les raisons suivantes:

- Architecture Structurée
 - Réutilisabilité des Composants
 - Flexibilité et Extensibilité
 - Performances et Sécurité

Symfony est optimisé pour la performance avec des outils de cache intégrés, et suit des pratiques de sécurité robustes.

of architecture et code : annexe C 3

```
explorateur ... └── UserController.php
    └── src/Controller
        └── User
            └── UserController.php
server-backed > src/Controller > UserController.php ...
1 ┌ <?php
2
3 namespace App\Controller;
4
5 use App\Entity\User;
6 use App\Utility\User;
7 use App\Utility\PasswordEncoder;
8 use App\Utility\.userService;
9 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
10 use Symfony\Component\HttpFoundation\Request;
11 use Symfony\Component\HttpFoundation\Response;
12 use Symfony\Component\Templating\EngineInterface;
13 use Symfony\Component\HttpFoundation\JsonResponse;
14 use Symfony\Component\HttpFoundation\RedirectResponse;
15
16 /**
17 * @Route("/user")
18 */
19 class UserController extends AbstractController
20 {
21     private $sservice;
22
23     /**
24      * La fonction ci-dessus est le constructeur en PHP qui initialise
25      * l'param user Service. Le paramtre « service » dans le code
26      * est une variable de classe qui nous permet d'accéder au service
27      * à partir de la méthode.
28
29     public function __construct(UserService $service)
30     {
31         $this->sservice = $service;
32     }
33
34     /**
35      * @param string $username
36      * @param string $password
37      *
38      * @return Response
39      */
40     public function loginAction(Request $request)
41     {
42         // ...
43     }
44
45     /**
46      * @param string $username
47      * @param string $password
48      *
49      * @return Response
50      */
51     public function registerAction(Request $request)
52     {
53         // ...
54     }
55
56     /**
57      * @param string $username
58      * @param string $password
59      *
60      * @return Response
61      */
62     public function changePasswordAction(Request $request)
63     {
64         // ...
65     }
66
67     /**
68      * @param string $username
69      * @param string $password
70      *
71      * @return Response
72      */
73     public function forgotPasswordAction(Request $request)
74     {
75         // ...
76     }
77
78     /**
79      * @param string $username
80      * @param string $password
81      *
82      * @return Response
83      */
84     public function resetPasswordAction(Request $request)
85     {
86         // ...
87     }
88
89     /**
90      * @param string $username
91      * @param string $password
92      *
93      * @return Response
94      */
95     public function changeEmailAction(Request $request)
96     {
97         // ...
98     }
99 }
```

La sécurité est la principale préoccupation lors de la conception et du développement d'une application. Elle revêt une importance cruciale pour plusieurs parties prenantes :

→ Pour l'utilisateur :

Toute faille de sécurité pouvant entraîner la fuite de données sensibles est inacceptable, car elle expose les utilisateurs à des risques de vol d'identité et de fraude.

→ Pour le client :

Une violation de la sécurité peut gravement affecter la réputation de l'entreprise. Les utilisateurs perdent confiance en une marque si leurs données ne sont pas sécurisées, ce qui peut entraîner des pertes économiques significatives et un recul de la fidélité des clients.

→ Pour le développeur :

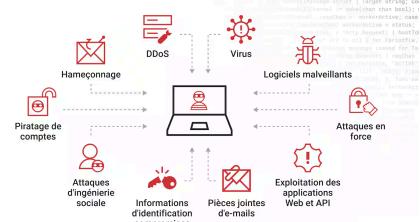
Un développeur est jugé sur sa capacité à créer des systèmes sécurisés et fiables. Une faille de sécurité peut non seulement ternir sa réputation, mais aussi entraîner des coûts supplémentaires en termes de temps et de ressources pour remettre le système en état.

La sécurité étant une préoccupation centrale, des mesures de sécurité rigoureuses et une vigilance constante indispensables pour prévenir les risques et assurer la stabilité et la fiabilité du système ont été prises.

➤ Exemples

Voici quelques-unes des attaques les plus fréquentes et les méthodes pour les prévenir :

- Injection SQL :
- Cross-Site Scripting (XSS) :
- Cross-Site Request Forgery (CSRF) :
- Brut Force
- Man-in-the-Middle (MITM)
- Déni de Service (DoS) et Dénie de Service Distribué (DDoS)
- Vol de Session



cf Détail dans l'annexe C.3.2.2

Si en tant que dev, je dois être vigilant et proactif pour prémunir mon application contre les attaques , **il est essentiel de pousser les utilisateurs et les clients à adopter de bonnes pratiques en matière de sécurité. Voici quelques exemples de ces bonnes pratiques :**

• Les Utilisateurs

- Utilisation de Mots de Passe Forts
- Authentification à Deux Facteurs (2FA)

• Les Clients

- Politiques de Sécurité Stricte
- Formation et Sensibilisation
- Audit et Surveillance

La sécurité va au-delà de la protection technique des systèmes et inclut la promotion de bonnes pratiques comme l'utilisation de mots de passe forts, l'authentification à deux facteurs, les mises à jour régulières, des politiques strictes, la formation continue et la surveillance active, pour prévenir les incidents, minimiser les risques et garantir la protection des données et la confiance des utilisateurs.



3) la

Couche Données



Avant de choisir et de configurer une base de données, il est crucial de bien comprendre les besoins de l'utilisateur et de l'application pour garantir une conception adaptée, supporter les fonctionnalités requises et offrir une performance optimale ; un rappel des besoins des utilisateurs à ce stade est donc essentiel.

D'autant plus que nous devons intégrer les contraintes liées à l'utilisation d'une base de données.

Identification des Besoins Utilisateur :

- **Scénarios d'Utilisation** : Analyser comment les utilisateurs interagiront avec l'application. Par exemple, quelles données ils devront consulter fréquemment, quelles informations ils devront saisir, et quelles actions ils devront effectuer.
- **Volumes de données** : Estimer la quantité de données que l'application devra gérer. Cela inclut le nombre d'utilisateurs, la fréquence des transactions, et la taille des enregistrements.
- **Performance et Réactivité** : Déterminer les exigences en matière de performance, comme les temps de réponse acceptables et la capacité à gérer des pics de charge.

Identification des Besoins de l'Application :

- **Fonctionnalités principales** : Recenser les fonctionnalités principales de l'application et identifier les données nécessaires pour les supporter. Par exemple, la gestion des utilisateurs, la création de tâches, la gestion de ces mêmes tâches, etc.
- **Intégration avec d'autres Systèmes** : Prendre en compte les besoins d'intégration avec d'autres systèmes ou services externes, tels que des API tierces ou des services de messagerie.
- **Sécurité et Conformité** : Évaluer les exigences en matière de sécurité des données et de conformité réglementaire (par exemple, GDPR pour les données personnelles).

Conception de la Base de Données

Une fois les besoins utilisateur et application bien compris, nous pouvons passer à la conception de la base de données. Pour cela, j'utilise la méthode **MERISE**

➤ **MERISE : Une Méthode d'Analyse et de Conception**

MERISE est une méthode française d'analyse et de conception des systèmes d'information, particulièrement utilisée pour la modélisation des données et des traitements. Développée dans les années 1980, elle était particulièrement pertinente pour l'informatisation des organisations. Bien que son utilisation ait diminué, ses concepts restent utiles, surtout pour la conception de bases de données relationnelles.

Analyse et Conception des Données :

Dans le cadre du titre de Concepteur Développeur d'Applications (CDA), nous nous concentrerons sur l'analyse et la conception des données relationnelles. **MERISE** propose trois niveaux d'analyse des données : **conceptuel**, **logique** et **physique**. Le but est de créer une base de données qui répond aux besoins de l'organisation en matière de stockage et de manipulation des informations.

Les Trois Niveaux de MERISE :

1. Niveau Conceptuel :

Modèle Conceptuel des Données (MCD) : À ce stade, on ignore les considérations techniques et se concentre sur les entités métiers et leurs relations. Les entités sont des abstractions des objets du monde réel manipulés par l'organisation (par exemple : utilisateurs, tâches principales, sous-tâches).

Objectifs : Identifier et organiser les informations sous forme d'entités et de propriétés sans se soucier des règles de gestion ou des types de données.

2. Niveau Logique :

Modèle Logique des Données (MLD) : Traduction du MCD en prenant en compte le type de SGBD (Système de Gestion de Bases de Données) utilisé. Les entités deviennent des tables, les propriétés des colonnes, et les identifiants des clés primaires. Les associations entre entités se transforment en clés étrangères.

Objectifs : Représenter les données de manière compatible avec le type de SGBD choisi (par exemple, relationnel), mais sans spécifier les détails d'implémentation.

3. Niveau Physique :

Modèle Physique des Données (MPD) : Traduction du MLD en instructions spécifiques au SGBD cible, généralement en SQL. Le MPD définit les types de données, les contraintes, les index et les colonnes calculées pour optimiser les performances.

Objectifs : Préparer le schéma de la base de données pour une implémentation concrète, incluant la création de tables et de relations physiques.

MERISE, avec ses trois niveaux d'analyse (conceptuel, logique, physique), offre une méthode structurée pour la conception des bases de données. Elle permet de transformer les besoins métier en une base de données optimisée et fiable, en passant par des étapes de modélisation qui garantissent la cohérence et la pertinence des données stockées.

❖ Inconvénients

Je ne décris pas **MERISE**, et bien que cette méthode soit puissante et structurée pour la conception et l'analyse des systèmes d'information, elle a aussi des inconvénients qu'il faut prendre en considération et anticiper.

Entre autres, cela inclut sa complexité, sa rigidité, sa courbe d'apprentissage, et sa pertinence réduite dans certains contextes modernes. Ces facteurs doivent être pris en compte lors de la décision d'utiliser Merise pour un projet particulier. Les équipes doivent évaluer si les avantages de Merise en termes de rigueur et de structure justifient les coûts et les efforts associés à son adoption.

J'utilise **MERISE** car cela est pour moi un excellent moyen de garder une ligne fixe et un code propre.

➤ Voyons les atouts qui m'ont aidé dans la conception du projet :

la Modélisation Conceptuelle des Données

- **Diagrammes Entité-Relation (ERD)** : Utiliser des diagrammes entité-relation pour représenter les entités principales et leurs relations. Cela permet de visualiser la structure des données de manière claire et organisée.
- **Attributs et Clés Primaires** : Définir les attributs de chaque entité et identifier les clés primaires, qui serviront d'identifiants uniques pour chaque enregistrement.

Le Choix du Système de Gestion de Bases de Données (SGBD) :

- **SGBD Relationnel vs. NoSQL** : En fonction des besoins identifiés, choisir entre un SGBD relationnel (comme MySQL ou PostgreSQL) ou un SGBD NoSQL (comme MongoDB ou Cassandra). Les bases de données relationnelles sont idéales pour les données structurées et les relations complexes, tandis que les bases de données NoSQL sont mieux adaptées pour les données non structurées et les grands volumes de données.

- **Critères de sélection :** Considérer des critères tels que la performance, la scalabilité, la facilité d'utilisation, la communauté de support, et les coûts.

La Normalisation des Données :

- **Élimination des redondances :** Structurer la base de données de manière à minimiser la duplication des données et à éviter les anomalies de mise à jour.
- **Optimisation des Requêtes :** S'assurer que les requêtes courantes peuvent être exécutées de manière efficace.

La Migrations et Scripts de Déploiement :

- **Gestion des Changements :** Utiliser des outils de migration de base de données pour gérer les changements de schéma de manière structurée et reproductible.
- **Automatisation :** Automatiser le déploiement des modifications de la base de données pour assurer la cohérence entre les environnements de développement, de test, et de production.

La Sécurité et Sauvegardes :

- **Contrôles d'accès :** Mettre en place des mécanismes de contrôle d'accès pour s'assurer que seules les personnes autorisées peuvent accéder ou modifier les données sensibles.
- **Chiffrement :** Utiliser le chiffrement pour protéger les données sensibles, tant au repos que en transit.
- **Sauvegardes Régulières :** Planifier et automatiser des sauvegardes régulières pour garantir la récupération des données en cas de perte ou de corruption.

La gestion des données commence par une analyse approfondie des besoins utilisateur et application, suivie par une conception soigneuse de la base de données. Cette approche garantit que la base de données sera bien adaptée pour supporter les fonctionnalités requises, offrir une performance optimale, et assurer la sécurité et l'intégrité des données ; bien que la sécurité soit en lien direct avec la couche métier.

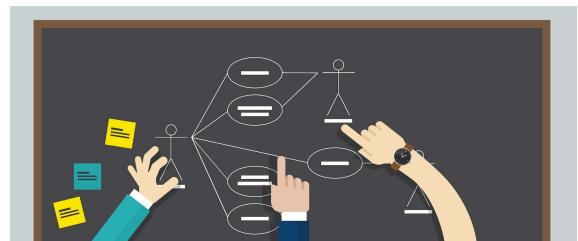
- Nous avons déjà analysé les besoins de l'utilisateur, nous allons les résumer dans des cas d'utilisation :
- Les **Use Case** écrit dans le langage **UML**.



Les modèles **UML** (Unified Modeling Language) sont des outils standardisés utilisés pour la modélisation et la conception de systèmes logiciels. Ils permettent de visualiser, spécifier, construire et documenter les artefacts d'un système de manière claire et structurée.

les modèles **UML** sont essentiels pour la conception et la documentation des systèmes logiciels complexes.

Ils améliorent la communication, garantissent une spécification précise, facilitent une conception robuste et offrent une documentation complète, ce qui est crucial pour le succès à long terme des projets logiciels.



Modèle Conceptuel des Données (MCD)

1. Les Use Case

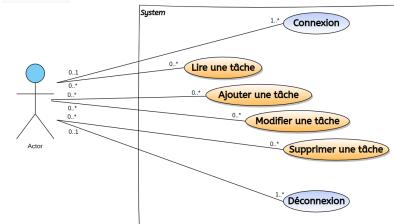
Les use cases clarifient les exigences fonctionnelles, améliorent la communication, centrent le développement sur l'utilisateur, facilitent les tests, et aident à prioriser les fonctionnalités et à planifier efficacement le projet.

> Exemple : TÂCHES PRINCIPALE :

- ★ Voici l'explication détaillée de chaque interaction de l'acteur avec le système, en tenant compte des multiplicités mentionnées :

1.1. Connexion (0..1 -> 1..*)

L'acteur peut se connecter au système une fois au maximum (0..1), mais chaque session de connexion peut impliquer plusieurs instances ou tentatives de connexion côté système (1..*).



cf diagramme : annexe D.4

Exemple : Un utilisateur peut se connecter à l'application (une seule fois à un instant donné), mais le système doit potentiellement plusieurs connexions simultanées ou des tentatives de connexion répétées de différents utilisateurs.

1.2. Déconnexion (0..1 -> 1..*)

L'acteur peut se déconnecter du système une fois au maximum (0..1), mais le système doit gérer potentiellement plusieurs déconnexions simultanées ou successives (1..*).

Exemple : Un utilisateur se déconnecte de l'application, mais le système doit traiter plusieurs déconnexions provenant de différents utilisateurs.

1.3. Gestion des tâches / Créer ; Lire ; Modifier ; Supprimer (0.. -> 0..)

L'acteur peut interagir avec plusieurs tâches de différentes manières (lecture, ajout, modification, suppression), et le système doit gérer ces multiples interactions (0..).

Exemple : Un utilisateur peut lire, ajouter, modifier et supprimer plusieurs tâches, et le système doit traiter chaque action en conséquence.

1.4. Le rôle des multiplicités

0..1 (acteur) : L'acteur peut réaliser l'action au maximum une fois dans une certaine période de temps.

1..* (système) : Le système doit être capable de gérer une ou plusieurs instances de l'action.

0..* (acteur) et (système) : L'acteur peut réaliser l'action un nombre indéfini de fois, et le système doit être capable de gérer un nombre indéfini d'instances de cette action.

2. Le Modèle Conceptuel de Données (MCD)

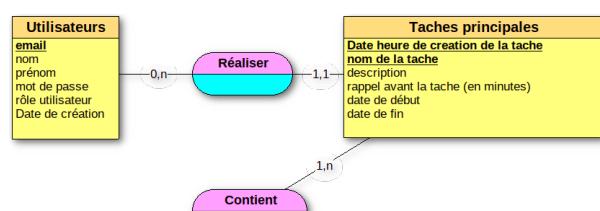
Le Use Case et le MCD sont deux outils complémentaires dans la conception d'un système.

Le Use Case décrit les interactions entre les utilisateurs et le système, tandis que le MCD représente la structure des données nécessaires pour soutenir ces interactions.

La gestion des Tâches principale sont définis comme ceci :

- **Use Case :** Les utilisateurs peuvent lire, ajouter, modifier, et supprimer des tâches.
- **MCD :** L'entité Tâche, avec ses attributs et sa relation avec l'entité Utilisateur, stocke les informations nécessaires pour ces opérations. Par exemple, pour ajouter une tâche, une nouvelle entrée est créée dans l'entité Tâche avec l'email de l'utilisateur correspondant. Pour modifier ou supprimer une tâche, l'email est utilisé pour identifier et effectuer les modifications nécessaires.

Détaillons un peu :



★ Mes Entité

J'ai deux entités :

1. L'entité **Utilisateurs**
2. L'entité **Tâches principales**

cf diagramme : annexe D.1

★ mes Propriétés

Chaque entité possède des propriétés simples et atomiques comme

- | | | |
|---------|----------------|----------|
| → email | → prénom | → ect .. |
| → nom | → mot de passe | |

❖ Nous avons :

1. Simplicité:

Une propriété doit représenter une seule valeur ou une seule information.

Exemple : Le nom d'un utilisateur est une propriété simple. Elle ne doit pas être décomposée en plusieurs sous-parties au niveau du MCD.

2. et Atomicité

Une propriété est dite atomique si elle ne peut pas être divisée en composants plus petits sans perdre sa signification. En d'autres termes, elle ne doit pas contenir plusieurs valeurs ou informations distinctes.

Exemple :

- Nom** : Plutôt que de stocker le nom complet d'une personne dans une seule propriété "NomComplet", il est préférable de le diviser en "Prénom" et "Nom".
- Date** : Une date est atomique car elle représente un point temporel unique (jour, mois, année).

★ Mes discriminants

Chaque entité doit avoir une propriété spécifique qui sert d'identifiant unique et discriminant pour garantir que chaque instance de l'entité est distincte et identifiable de manière unique. Cela assure l'intégrité, l'efficacité et la clarté dans la gestion et la manipulation des données au sein de la base de données.

Dans mon entité '**Utilisateurs**' , la propriété '**email**' sert de discriminant car elle ne peut avoir plusieurs '**email**' identiques → chaque utilisateur possède **un email** alors que plusieurs utilisateurs peuvent avoir **le même nom, le même prénom voir les deux**.

Il existe **plusieurs 'Eric' 'Corbisier'** mais **un seul** à l'adresse email 'eric@corbisier.fr'.

Utilisateurs
email
nom
prénom
mot de passe
rôle utilisateur
Date de création

❖ Un Cas particulier

Utiliser plusieurs propriétés pour générer un identifiant composite est parfois nécessaire pour garantir l'unicité et l'intégrité des données lorsque aucune propriété unique ne suffit.



C'est le cas de mon entité '**Tâches principales**' :

Tâches principales
Date heure de création de la tâche
<u>nom de la tâche</u>
description
rappel avant la tâche (en minutes)
date de début
date de fin

- Plusieurs tâches peuvent avoir le même nom
- Plusieurs tâches peuvent avoir le même horodatage de création

Pour cela '**Date heure de création de la tâche**' et '**nom de la tâche**' vont générer un discriminant unique.

★ **Les associations**

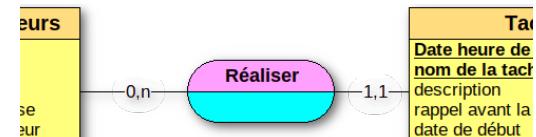
En examinant notre modèle, nous remarquons des liens entre les entités :

- **les liens sémantiques.**

Un lien sémantique est une relation significative entre deux entités, souvent exprimée par un verbe, qui clarifie leurs rôles et interactions bidirectionnelles.

Dans mon projet Tranquillo

1. un utilisateur "**réalise**" une tâche
2. et une tâche "**est réalisée**" par un utilisateur.



★ **Les Cardinalités**

Nous les avons déjà abordées dans le sujet **UML**. Dans mon projet cela signifie que :

- **Chaque utilisateur** peut **réaliser plusieurs tâches principales**, mais il n'est **pas obligatoire qu'un utilisateur ait des tâches principales à réaliser** (cardinalité 0,n).
- **Chaque tâche principale** doit **être réalisée par un utilisateur unique**, ce qui signifie qu'elle ne peut être **associée qu'à un seul utilisateur** (cardinalité 1,1).

L'analyse des données au niveau logique aboutit à la création d'un Modèle Logique des Données (MLD), qui est une traduction du Modèle Conceptuel des Données (MCD).

À ce stade, il est nécessaire de prendre en compte le type de Système de Gestion de Base de Données (SGBD) que l'on va utiliser. Le type de SGBD influence la manière dont les données seront représentées et organisées (tables, réseaux, hiérarchies, documents).

Nous devons néanmoins choisir le type de SGBD.

3. **Les types de base de données**

3.1. **Le modèle relationnel**

Le modèle relationnel, introduit par E. Codd en 1970, est basé sur douze règles théoriques et le concept de relation, défini comme un ensemble de **n-uplets** (*cfr chapitre Définitions*) représentant des propriétés spécifiques d'un objet.

Il assure une gestion précise des contraintes d'intégrité pour la cohérence des données. Les **systèmes de gestion de bases de données relationnelles (SGBDR)** utilisent le SQL pour effectuer toutes les opérations nécessaires, y compris la gestion des transactions.

Les transactions dans ces systèmes respectent les propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité), garantissant ainsi que

- toutes les modifications sont soit entièrement réalisées, soit annulées (atomicité),
- les données restent cohérentes,
- les modifications ne sont visibles que par l'utilisateur exécutant la transaction (isolation),
- et que les effets des transactions sont permanents, même en cas de défaillance système (durabilité).

3.2. Le modèle NoSQL

Les bases de données NoSQL sont souvent présentées comme une solution idéale, mais plusieurs points doivent être considérés avant de choisir cette option.

Les systèmes NoSQL offrent un gain de performances en répartissant les données sur plusieurs serveurs, facilitant ainsi la montée en charge avec l'ajout de nouveaux serveurs et permettant l'utilisation d'un schéma flexible.

Cependant, ils présentent des inconvénients tels que la complexité des requêtes déplacée vers la logique de l'application et, dans certains cas, le non-respect des propriétés ACID, nécessitant un effort supplémentaire pour assurer la cohérence des données.

3.3. Mon choix

Pour les bases de données relationnelles*, comme celle que j'ai choisie, les données sont organisées en tables représentant les entités et relations définies dans le MCD, adaptées aux contraintes du SGBD-R choisi.

Ma base de données respecte les propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité), garantissant la fiabilité des transactions, l'intégrité des données, l'isolation des modifications jusqu'à validation, et la durabilité des enregistrements.



(* **SGBD-R** - systèmes de gestions de base de données relationnelle)

Le modèle Logique de Données

Pour le moment, nous ne nous préoccupons pas de l'implémentation spécifique, ni du choix de l'éditeur ou du SGBD-R, et nous restons au niveau conceptuel sans entrer dans les détails du typage des données pour assurer la compatibilité avec tous les SGBD-R.

Si le modèle logique ne contient que des concepts généraux et communs à tous les SGBD-R, c'est justement dans le but de pouvoir l'utiliser quelque soit le choix de notre base de données.



Nous allons avant tout effectuer une transformation.

La transformation du MCD en MLD-R implique la conversion suivante:

- | | |
|---|--|
| <ul style="list-style-type: none">→ des entités→ des propriétés→ des identifiants→ et des associations | <ul style="list-style-type: none">→ en tables,→ en colonnes,→ en clés primaires,→ en clés étrangères. |
|---|--|

1. Les tables et les colonnes (y compris les clés primaires) **doivent avoir des noms uniques**, respectant les mêmes règles d'unicité qu'au niveau conceptuel.
2. Chaque table doit avoir **une clé primaire**.

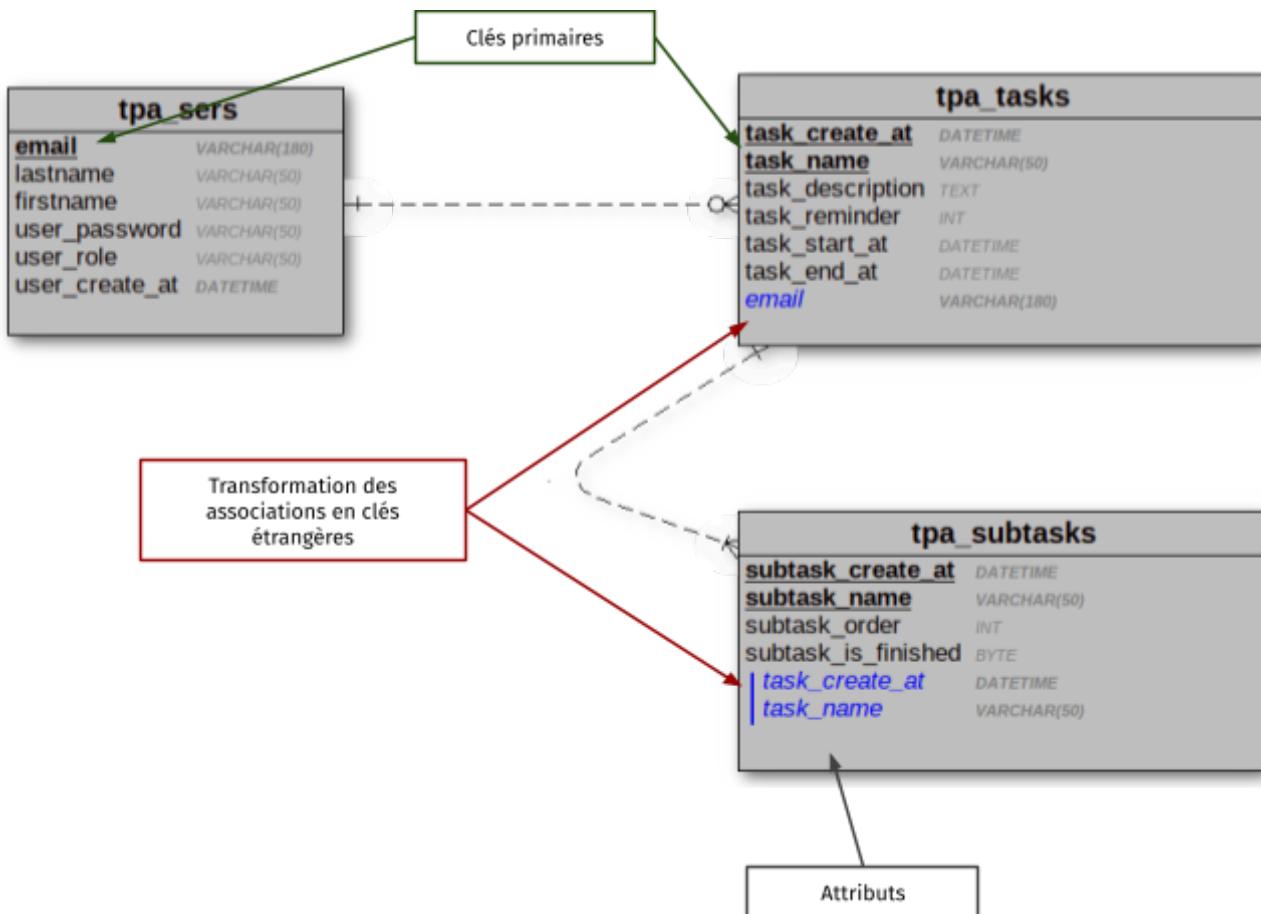
Nous allons avoir les notions de

- Contraintes
- Unicité
- Attributs
- Clés secondaires

- Clés étrangères

- Dénormalisation

Voici la représentation **UML** de mon **MLD** :



Nous retrouvons les mêmes particularités dans le MLD que dans le MCD :

La clé primaire doit être '**UNIQUE**'. Par conséquent, elle peut être représentée par une seule colonne, comme dans la table **tpa_users** ou par deux colonnes comme dans la table **tpa-tasks** (cf sous-titre 'discriminant' dans le chapitre "Modèle Conceptuel des Données").

Connaissant la cible (cf page suivante), on peut, et même on doit, spécifier les types de données pour chaque colonne selon les types, leurs précisions (longueurs de chaînes, tailles pour les entiers et parties décimales...) et les contraintes (non nullité, unicité).

On peut également se préoccuper des performances et ajouter des indexées et même des colonnes calculées (pour stocker une bonne fois pour toute un total par exemple, ce qui évite de le calculer à chaque besoin)

Pour cela, il me faut :

★ savoir où déclarer les clés étrangères :

Cardinalité des Associations :

On se concentre sur les cardinalités maximales des associations pour déterminer où placer les clés primaires et les propriétés associées. Selon ces cardinalités, les clés peuvent être ajoutées dans une des deux tables impliquées ou dans une nouvelle table de jointure.

1. Associations 1/n :

Si une association a une cardinalité maximale de '**1**' d'un côté et '**n**' de l'autre, la clé primaire de la table du côté '**1**' devient une clé étrangère dans la table du côté '**n**'. Par exemple, si une tâche appartient à un utilisateur (**1/n**), la clé primaire de la catégorie devient une clé étrangère dans la table des tâches.

2. Associations Réflexives 1/n :

Pour les associations réflexives '**1/n**', la clé primaire est également déplacée vers le côté '**1**', c'est-à-dire dans la même table, ajoutant ainsi une clé étrangère à cette table.

3. Associations n/n :

Dans une association où les cardinalités maximales sont '**n**' des deux côtés, les clés primaires des deux tables sont déplacées vers une nouvelle table de jointure, où elles deviennent des clés étrangères. La combinaison de ces clés étrangères forme la clé primaire de la table de jointure.

4. Propriétés des Associations :

Dans une association '**1/n**', les propriétés de l'association sont ajoutées à la table contenant les clés étrangères. Dans une association '**n/n**', ces propriétés sont placées dans la table de jointure.

5. Associations Ternaires :

Pour les associations ternaires (impliquant plus de deux entités), les clés primaires de toutes les entités sont placées dans une nouvelle table de jointure, où elles deviennent des clés étrangères, et leur combinaison forme la clé primaire de la table de jointure, avec les propriétés de l'association le cas échéant.

6. Nommage des Tables de Jointure :

Le nom des tables de jointure est basé soit sur le verbe de l'association, soit sur la concaténation des noms des entités concernées. Par exemple, "**tpa_tasks_tpa_users**" pour une association entre "**tpa_task**" et "**tpa_users**". Il est crucial d'éviter les ambiguïtés et les noms trop longs.

Pour convertir notre MLD en MPD, nous devons choisir une base de données pour générer le script adapté.

J'utilise **MariaDB** avec **SQL** car, en plus d'être fourni avec mon serveur PHP, cette BDD remplit les conditions pour mon application et respecte les propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité).



Le modèle Physique de Données

Le **MPD (Modèle Physique des Données)** est la dernière étape, avant implémentation, de la conception des données, traduite en scripts SQL, appelés **DDL (Data Definition Language)**, pour définir ou modifier la structure des tables et des relations d'une base de données relationnelle.



Nous allons aborder une nouvelle notion, même si elle a déjà vaguement été citée dans le chapitre MLD :

➤ La Dénormalisation et le clé secondaire

L'ajout et la suppression de colonnes, bien que rares, sont des actions de dénormalisation pour optimiser les performances en mémoire et temps de traitement, répondant aux besoins spécifiques de l'organisation malgré des compromis sur certaines bonnes pratiques de conception.

★ Exemple

L'utilisation de clés primaires numériques auto-incrémentées dans les bases de données relationnelles simplifie les requêtes, assure l'uniformité et améliore les performances des jointures.

❖ Point de vigilance

● Modifications et Impacts

Les modifications apportées au MPD, telles que l'introduction de clés primaires techniques, nécessitent des ajustements importants. Les identifiants définis dans le MCD et les clés primaires du MLD-R sont essentiels et ne peuvent pas être supprimés.

Lors de l'ajout d'une **clé primaire technique** :

- l'identifiant d'origine doit être
 - déclaré **unique**
 - et devenir une **clé secondaire**,
- les clés étrangères doivent être
 - ajustées pour référencer les nouvelles clés primaires techniques.

● Conséquences sur le Mapping Applicatif

Ces changements impactent non seulement le MPD, mais aussi le mapping dans les applications (par exemple, les ORM). Il est crucial de prévoir et gérer ces impacts pour maintenir la cohérence et la fonctionnalité des données au niveau applicatif.

➤ Cas Concret : MPD pour MariaDB en SQL pour Tranquillo

Nous pouvons découvrir l'apparition d'**ID techniques** qui deviennent **clés primaires** :

- "user_id"
- "task_id"

Utiliser des **ID techniques**, numériques et souvent auto-incrémentés, comme clés primaires dans une base de données assure unicité, simplicité des requêtes, et amélioration des performances grâce à une indexation efficace.

Les **discriminants** du MCD deviennent par conséquent des **clés secondaires** recevant la contrainte "**UNIQUE**".

→ Dans la table "**tpa_users**" :

- La clé secondaire est la colonne avec l'attribut "**email**", nommée "**tpa_users_ukey**".

→ Dans la table "**tpa_tasks**" :

- La clé secondaire est une contrainte unique sur les colonnes
 - ◆ "**task_name**" et
 - ◆ "**task_create_at**",
- nommée "**tpa_tasks_ukey**".

Cela permet de maintenir la cohérence, la lisibilité et la propreté du script SQL tout en respectant nos modèles de conception et logique de données.

```
CREATE TABLE tpa_users
IF NOT EXISTS tpa_users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(180) NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    firstname VARCHAR(50) NOT NULL,
    user_password VARCHAR(255) NOT NULL,
    user_role VARCHAR(255) NULL);
```

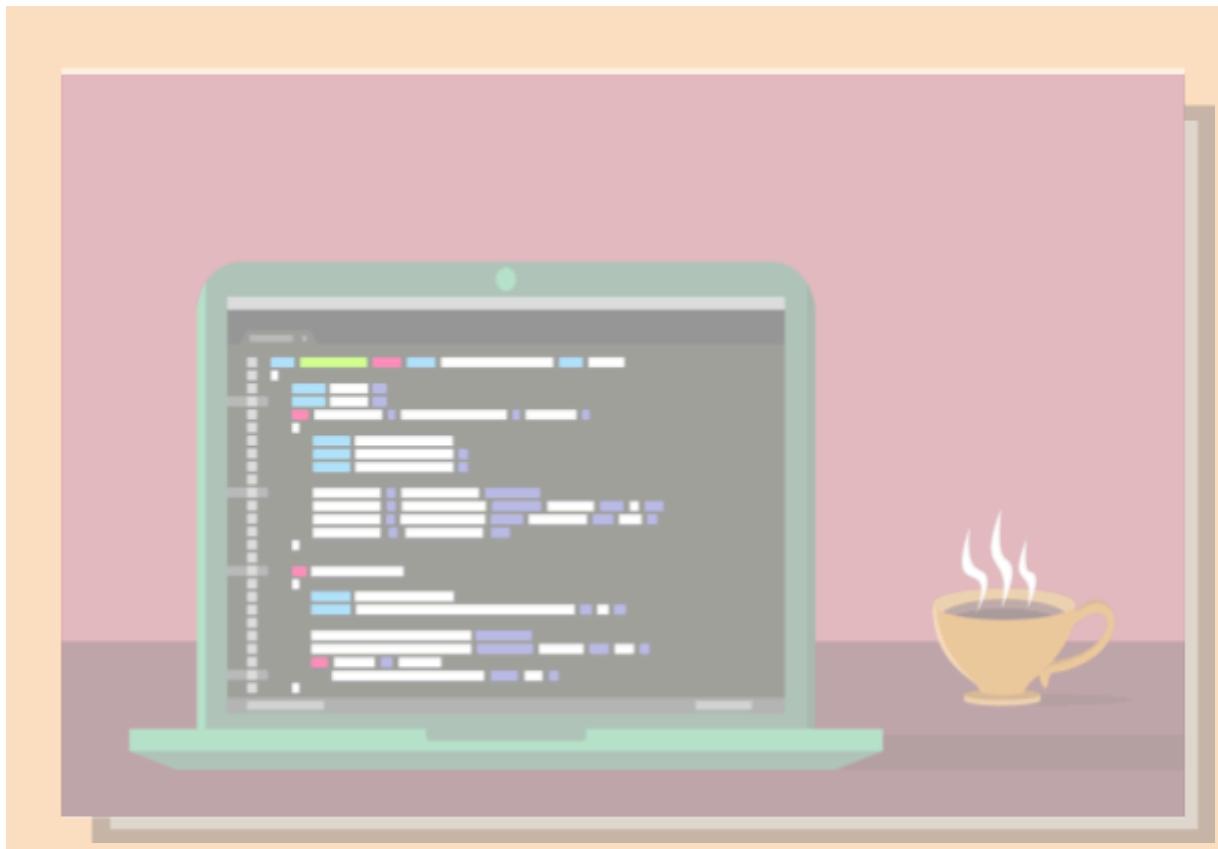
#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut
1	task_id	int(11)			Non	Aucun(e)
2	task_name	varchar(50) utf8mb4_general_ci			Non	Aucun(e)
3	task_description	text utf8mb4_general_ci			Oui	NULL
4	task_reminder	int(11)			Oui	NULL

Action	Nom de l'index	Type	Unique	Comprimé	Colonne
Éditer	Renommer	Supprimer	PRIMARY	BTREE	Oui
Éditer	Renommer	Supprimer	tpa_tasks_ukey	BTREE	Oui
Éditer	Renommer	Supprimer	tpa_tasks_users_fkey	BTREE	Non

cf Définitions : Entêtes dans les Fichiers SQL

le

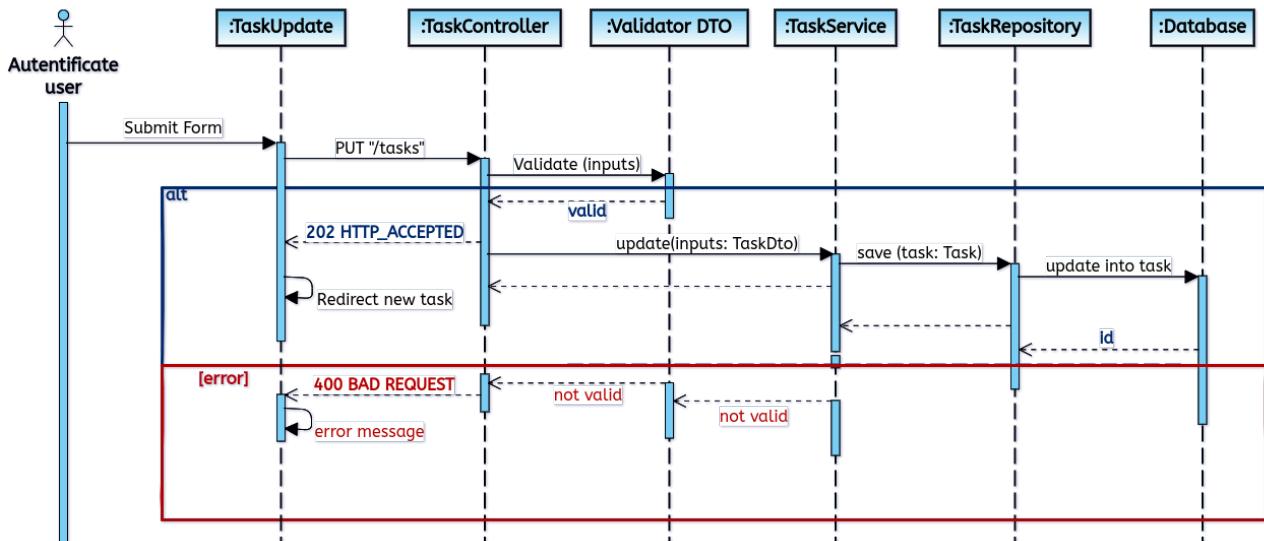
Code (Réalisation)



Pour compléter la conception, nous allons utiliser les diagrammes de séquences :

Le diagramme de séquence, outil essentiel en modélisation UML, visualise les interactions dynamiques entre objets d'un système, détaillant les messages échangés pour illustrer les processus et relations, aidant à comprendre, documenter, et optimiser les cas d'utilisation complexes et les flux de travail.

> **Voici le diagramme d'un utilisateur qui modifie une de ces tâches :**



Nous pouvons visualiser :

- ★ chaque étape entre le moment où l'utilisateur soumet le formulaire et celui où les données sont enregistrées en base de données.
- ★ les réponses générées à chaque étape de validation, qu'elles soient positives ou négatives.

Cela nous permet de préparer notre couche métier pour qu'elle puisse traiter les informations (comme notre formulaire de modification de tâche), en passant par toutes les validations nécessaires.

- ★ Si toutes les validations sont réussies,
 - les données sont sauvegardées en base de données,
 - et une réponse positive (**code 202 HTTP OK**) est envoyée au client ;
- ★ Sinon, une réponse négative (**code 400 HTTP BAD_REQUEST**) est retournée.

Structure de l'API (Couche métier)

L'API est découpé en composants ayant des usages spécifiques :

<input type="checkbox"/> Récupérer les données	CONTROLLER
<input type="checkbox"/> Validées ses données	DTO
<input type="checkbox"/> Appliqué les bons process	SERVICES
<input type="checkbox"/> Persister les données	ENTITY et REPOSITORY



cf code API : annexe C.3.2

1. Controller :

Le contrôleur gère les requêtes HTTP entrantes et les réponses sortantes, servant d'interface entre le client (comme un navigateur ou une application mobile) et le service de l'application.

- **Traitement des Requêtes** : Reçoit les requêtes HTTP (GET, POST, PUT, DELETE) et les dirige vers les services appropriés.
- **Validation Initiale** : Peut effectuer une validation initiale des données avant de les passer au service.
- **Retour de Réponses** : Prépare et envoie les réponses HTTP appropriées au client, incluant les codes de statut et les messages.

2. Validateur DTO (Data Transfer Object) :

Le Validateur DTO vérifie que les données transférées entre le client et l'application respectent les contraintes définies avant d'être traitées par les services.

- **Validation des Données** : Vérifie que les données envoyées par le client sont complètes, correctes et respectent les règles de validation (par exemple, formats d'email, longueurs de champ).
- **Sanitization** : Assure que les données sont nettoyées et sécurisées avant traitement.
- **Erreur de validation** : Retourne des erreurs spécifiques si les données ne respectent pas les contraintes définies, souvent avant même que les données atteignent le service.

3. Service :

Le service encapsule la logique métier de l'application. Il contient les règles de traitement des données et les opérations qui doivent être exécutées.

- **Logique métier** : Implémente les règles et processus métier (par exemple, calculs, vérifications, transformations de données).
- **Coordination** : Interagit avec le repository pour effectuer les opérations de lecture et d'écriture en base de données.
- **Gestion des Transactions** : Peut gérer les transactions pour s'assurer que les opérations sur la base de données sont atomiques et cohérentes.

4. Repository :

Le repository est responsable de l'interaction directe avec la base de données. Il effectue les opérations de persistance CRUD.

- **Accès aux Données** : Fournit une interface pour accéder et manipuler les entités en base de données.
- **Abstraction** : Abstrait la logique d'accès aux données de la logique métier, facilitant les changements de technologie de stockage.
- **Requêtes Complexes** : Peut implémenter des requêtes complexes pour récupérer des données spécifiques de la base de données.

Symfony utilise l'ORM Doctrine pour simplifier la manipulation des données avec des objets PHP au lieu de requêtes SQL, gérant automatiquement les opérations CRUD et assurant l'intégrité des données, ce qui permet aux développeurs de se concentrer sur la logique métier.

✓ serveur-backend
> bin
> config
> migrations
> public
✓ src
> ApiResource
✓ Controller
◆ .gitignore
♫ HomeController.php
♫ TaskAdminController.php
♫ TaskController.php
♫ UserAdminController.php
♫ UserController.php
> DataFixtures
> Dto
> Entity
> EventListener
> Helper
> Repository
> Security
> Service
> Validator
♫ Kernel.php
> templates
> tests
> translations
> var
> vendor
☰ .editorconfig
⚙ .env
☰ .env.local
☰ .env.test

ORM (Object-Relational Mapping)

En utilisant un ORM, on bénéficie d'une abstraction et d'une simplification significatives de l'interaction avec la base de données. Cela permet de se concentrer sur la logique métier tout en assurant une gestion cohérente et performante des données.

Nous *limiterons son utilisation aux fonctions CRUD* pour maintenir la simplicité, la clarté et l'efficacité du code mais surtout pour permettre notre objectif principale : *la séparation des responsabilités*.

```
database > sql > 01_create_user_api.sql > ...
> Run | New Tab | Active Connection
1 CREATE USER 'api'@'%' IDENTIFIED BY 'password';
> Run | New Tab
2 GRANT SELECT, INSERT, UPDATE, DELETE ON `tranquillo`.* TO
'api'@'%';
```

cf annexe C.4.2

★ Détailons cela :

➤ Simplicité et Clarté :

- En limitant l'usage de l'ORM aux opérations CRUD (Create, Read, Update, Delete), on simplifie l'architecture et rend le code plus clair et plus prévisible.
 - **Facilité de compréhension** : Les développeurs peuvent facilement comprendre et suivre les interactions avec la base de données.
 - **Réduction de la Complexité** : Évite la prolifération de requêtes SQL complexes et spécifiques dans le code de l'application.

➤ Séparation des responsabilités :

- En restreignant l'accès aux fonctions CRUD, on maintient une séparation claire entre la logique métier et la logique d'accès aux données.
 - **Modularité** : Facilite la modularisation et le test de la logique métier indépendamment de l'implémentation des opérations de base de données.
 - **Réutilisabilité** : Permet de réutiliser les mêmes opérations CRUD à travers différents services ou parties de l'application.

➤ Performance et Optimisation :

- Les opérations CRUD sont souvent optimisées par les ORM pour des performances maximales.
 - **Optimisation automatique** : L'ORM peut appliquer des optimisations automatiques pour les opérations CRUD, telles que le caching et le pooling de connexions.
 - **Efficacité des Requêtes** : Les ORM sont conçus pour générer des requêtes SQL optimisées pour les opérations CRUD, réduisant ainsi le risque de mauvaises performances dues à des requêtes mal écrites.

➤ Voici un exemple de l'utilisation de l'ORM Doctrine :

cf annexe C.3.2.6

Chaque '**'attribut'** (colonne) de la table **tpa_tasks** de la base de données est associée à une variable privée de la classe **Task** de notre entité

Si le nom d'une colonne de la base de données devait être modifié, seule la classe **Task** serait affectée, sans impacter toutes les classes qui utilisent l'Entity Manager pour accéder à ces variables. Cela centralise les modifications et réduit les risques d'erreurs dans l'ensemble du code.

```
9  use Doctrine\ORM\Mapping as ORM;
10 use Symfony\Component\Serializer\Annotation\Groups;
11 use Symfony\Component\Validator\Constraints as Assert;
12
13 #[ORM\Entity(repositoryClass: TaskRepository::class)]
14 #[ORM\Table(name: "tpa_tasks")]
15 #[ORM\Index(name: "tpa_users_id_tasks_ikey", columns: ["users_id"])]
16 #[ORM\Index(name: "tpa_tasks_name_ikey", columns: ["task_name"])]
17 #[ORM\Index(name: "tpa_tasks_create_at_ikey", columns: ["task_create_at"])]
18 #[ApiResource]
19 class Task
20 {
21     public function __construct()
22     {
23     }
24
25     #[ORM\Id]
26     #[ORM\GeneratedValue]
27     #[ORM\Column(name: "task_id")]
28     #[Assert\PositiveOrZero()]
29     #[UserRegex(regex: 'number', entity: "task", field: "id")]
30     private ?int $id = null;
31 }
```

On ne peut parler 'Code' sans parler des classes. Aussi bien pour l'interface utilisateur que la couche métier utilisent des classes.

Et pour cause, la base même du CDA est l'orienté Objet et les classes en sont un fondement..

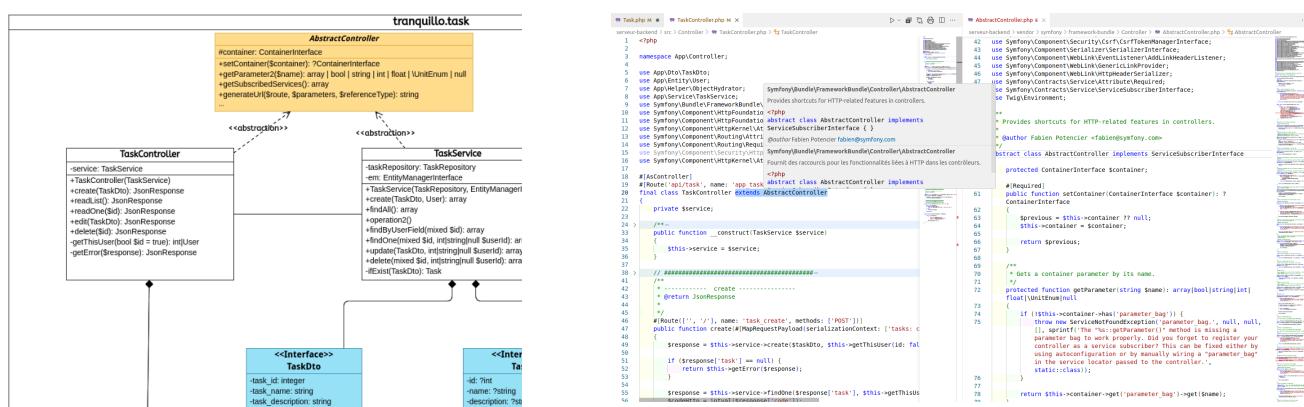
Les classes sont des structures fondamentales en programmation orientée objet qui définissent des modèles pour créer des objets, incluant des propriétés (attributs) et des fonctions (méthodes) :

- Les méthodes sont des fonctions définies à l'intérieur des classes qui décrivent les comportements des objets.
- Les classes abstraites ne peuvent pas être instanciées directement et sont conçues pour être étendues par d'autres classes, utilisant le mot-clé '**abstract**' pour indiquer qu'une méthode doit être implémentée dans les sous-classes.
- Le mot-clé '**extend**' permet à une classe de hériter des propriétés et méthodes d'une autre classe, favorisant la réutilisation du code.
- Les modificateurs d'accès '**public**' et '**private**' contrôlent la visibilité des propriétés et méthodes :
 - '**public**' permettant l'accès depuis l'extérieur de la classe, tandis que
 - '**private**' limite l'accès à l'intérieur de la classe uniquement, renforçant l'encapsulation et la protection des données.

Pour parvenir à créer et/ou utiliser des classes de manière optimale, nous nous sommes basés sur un outil de modélisation en programmation orientée objet (**diagramme de classe**) qui permet de visualiser les structures des classes, leurs attributs, méthodes et les relations entre elles.

Il aide à concevoir et documenter la structure d'un système logiciel, facilitant ainsi la compréhension et la communication entre les membres de l'équipe de développement.

Il s'agit du diagramme de classe :



cf annexe D.3

1. Validation des données

La validation des données se fait sur les couches Utilisateur et Métier. Nous nous assurons que :

- Un '**nom**' doit être un texte sans caractères spéciaux
- Un '**email**' doit répondre au format email
- Une '**date**' au format Date
- Les **messages d'erreur** sont clairs et utiles à l'utilisateur tout en évitant de divulguer des informations sensibles.

Interface

```
tranquilloapp > app > lib > packages > ts Patterns.js
1 import { getType, printR } from './functions';
2 import { localize } from './localize';
3
4 class Pattern {
5   constructor(pattern) {
6     public async fieldValue(string, fieldName: string): Promise<string | null> {
7       const namePattern: RegExp = `/^${pattern}/`;
8       if (!namePattern.test(fieldName)) {
9         return printR(localize(`pattern.name`), fieldName);
10      }
11      return null;
12    }
13
14    public async email(email: string): Promise<string | null> {
15      const emailPattern: RegExp = `/^${pattern}/`;
16      if (!emailPattern.test(email.trim())) {
17        return localize(`pattern.email`);
18      }
19      return null;
20    }
21
22    public async password(password: string): Promise<string | null> {
23      const passPattern = `/^${pattern}/`;
24      if (!passPattern.test(password.trim())) {
25        return localize(`pattern.password.length`);
26      }
27      if (password.length === 0) {
28        return localize(`pattern.password.force`);
29      }
30      if (!passwordPattern.test(password.trim())) {
31        return localize(`pattern.password.force`);
32      }
33      return null;
34    }
}
```

Métier (API)

```
server-backend > src > Dio > UserDio.php
12
13 class UserDio implements UserInterface, PasswordAuthenticatedUserInterface
14 {
15   /**
16    * @userRegexp(regex: "number", entity: "user", field: "id")
17    */
18   #groups["users: read", "users: create", "tasks: all"]
19
20   private ?int $id = null;
21
22   /**
23    * @assertEmail(message: "error.user.email: Invalid email")
24   */
25   #assertEmailBlank(message: "error.user.email: The key <email> must be a non-empty string")
26   #tpaking(maxIn: 3, max: 100, entity: "user", field: "email")
27   #groups["users: read", "users: create", "users: put"]
28
29   private ?string $email = null;
30
31   /**
32    * @var string The user roles
33   */
34   #groups["users: read", "user.admin: put"]
35   private array $roles = [];
36
37   /**
38    * @var string The hashed password
39   */
40   #userRegexp(regex: "password", entity: "user", field: "password")
41   #assertEmailBlank(message: "error.user.password: The key <password> must be a non-empty string")
42   #tpaking(maxIn: 3, max: 50, entity: "user", field: "password")
43   #groups["users: create", "users: pass"]
44
45   private ?string $password = null;
46 }
```

2. Métier (API)

cf annexe C.3.2.2

- Implémenter une limitation, de type Rate Limiting, du taux de requêtes pour prévenir les abus et les attaques par déni de service (DoS).
- Désactivation des méthodes HTTP non utilisées et acceptées aux seules nécessaires (par exemple, désactiver TRACE, OPTIONS si non utilisées).
- Détection d'intrusion : Mettre en place des mécanismes pour détecter et alerter en cas de tentatives d'intrusion.

3. Sécurisation des données

- Encodage des caractères spéciaux pour éviter les injection SQL ou XSS (*cf chapitre 'Importance des Bonnes Pratiques'*)
- Mise en place de Content Security Policy (CSP) : Utiliser CSP pour empêcher les attaques de type XSS en contrôlant les ressources que le navigateur peut charger pour une page donnée
- Utilisation de HTTP Security Headers en ajoutant des en-têtes de sécurité HTTP comme Strict-Transport-Security, X-Content-Type-Options, X-Frame-Options, et X-XSS-Protection.
- Communication en HTTPS en production
- Hachage du mot de passe de manière à la rendre inutilisable par un pirate si la base de données est corrompue et lire par des personnes malveillantes.

Ma classe 'User' implémente le 'PasswordAuthenticatedUserInterface' de Symfony.

er_id	email	lastname	firstname	user_password	user
25	contact@corbisier.fr	CORBISIER	Eric	\$2y\$13\$Z1aOGECmKSAJzPILyNvEd80rGsmV5gDRyZ3UGK0... [!ROI]	
26	ecorbisier.simplon@gmail.com	CORBISIER	Eric	\$2y\$13\$QB3TNolNIUrJ2HCVIKtVeOfh7AcG3WsYR50Ln/23UP... [!ROI]	

cf annexe C.4.4 et C.4.5

- Utilisation de jeton de type JWT pour sécurisé l'authentification de l'utilisateur entre l'API et son interface d'utilisation

- Token JWT
- Token refresh du JWT
- Token CSRF (cf chapitre 'veille sécurité')

Interface

```
tranquilloApp > app > stores > ts task.ts > TaskStore > loadPage
8 class TaskStore {
18 }
19
20 subscribe = (action: any) => {
21   return this.tasks.subscribe(action);
22 };
23
24 private async loadPage(page: number, user_token: string): Promise<Task[]> {
25   let url = `/task`;
26   let response = await client.sendRequest<TaskResponse>(
27     url,
28     {
29       method: "get",
30       user_token
31     }
32   );
33   this.user_token = user_token;
34
35   const task: Task[] = response.task || [];
36   if (!task.length) {
37     this.no_more_results = true;
38   }
39   this.tasks.set(response.task);
40 }
41
42 loadNextPage() {
43   if (this.no_more_results) return;
44   return this.loadPage(this.page + 1, this.user_token);
45 }
46
47 private async createOrUpdate(
48   task: Task,
49   user_token: string,
50   method: string = "post"
51 ) {
52 }
```

cf annexe C.1.3

Métier (API)

- Implémenter un contrôle d'accès basé sur les rôles pour gérer les permissions des utilisateurs.

```
serveur-backend > src > EventListener > UsersJWTCreatedListener.php > ...
14 final class UsersJWTCreatedListener extends AbstractController
15 {
16   /**
17    * @return void
18   */
19   public function onJWTCreated(JWTCreatedEvent $event)
20   {
21     $request = $event->getRequestStack()->getCurrentRequest();
22
23     // Récupérer le User-Agent de la requête client
24     $userAgent = $request->headers->get('User-Agent');
25
26     // Récupérer les données et les en-têtes du JWT
27     $payload = $event->getData();
28     $header = $event->getHeader();
29
30     // Ajouter le User-Agent au payload si vous en avez besoin
31     $payload['ip'] = $request->getClientIp();
32     // Ajouter le User-Agent au header du JWT
33     $payload['city'] = "JWTTranquillo";
34     $payload['User-Agent'] = $userAgent;
35
36     // Définir les nouvelles données et en-têtes pour le JWT
37     $event->setData($payload);
38     // $event->setHeader($header);
39
40     // Définir la date d'expiration du JWT
41     $expiration = new \DateTime('+30 day');
42     $expiration->setTime(2, 0, 0);
43     $payload['exp'] = $expiration->getTimestamp();
44     $event->setData($payload);
45
46   }
47
48 }
```

cf annexe C.3.2.1

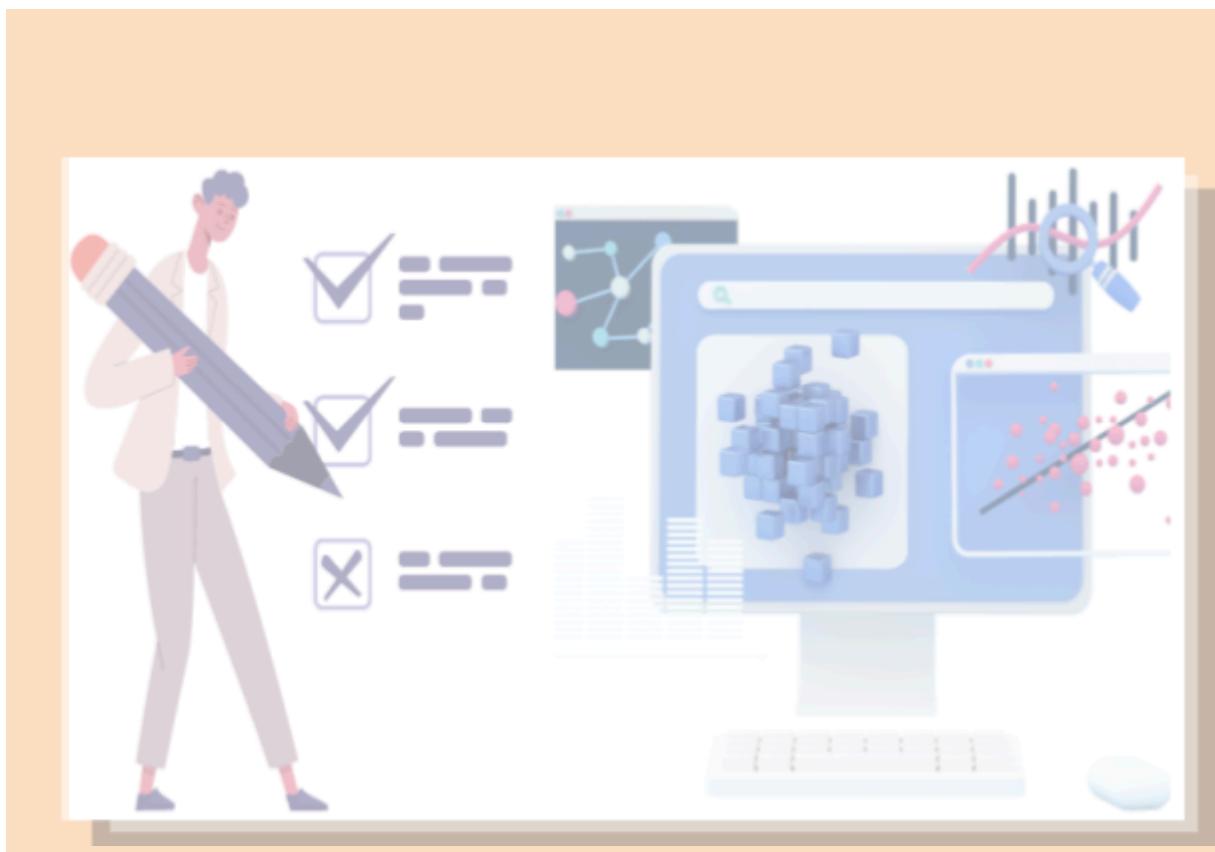
Bien que tout ne soit pas encore totalement mis en œuvre, je déploie des efforts constants et rigoureux pour atteindre cet objectif.

Ma priorité est d'assurer à mes utilisateurs une expérience d'utilisation de l'application qui soit à la fois fluide et sécurisée.



les

Tests



L'utilisation de différents types de tests est cruciale pour assurer la qualité, la fiabilité et la performance d'une application. Les tests unitaires permettent de détecter les erreurs.

- Au niveau des fonctions individuelles, les tests d'intégration s'assurent que les modules fonctionnent ensemble correctement,
- Les tests fonctionnels valident que l'application répond aux besoins des utilisateurs, et
- Les tests de performance garantissent que l'application peut supporter la charge attendue.

En combinant ces tests, nous pouvons livrer un produit de haute qualité qui répond aux attentes de nos utilisateurs et clients.



Pendant mon apprentissage, j'ai réalisé que mettre en place les tests dès le début du développement apporte de nombreux avantages, notamment la détection précoce des bugs, l'assurance de la qualité du code, la facilitation du refactoring, la documentation du comportement du code, la préparation à la livraison continue, et une meilleure collaboration en équipe.

Adopter une approche de test dès le début du cycle de développement est une stratégie proactive qui conduit à des résultats de meilleure qualité et à une satisfaction accrue des utilisateurs finaux.

Pour compenser les tests automatisés en cours de réalisation, j'ai utilisé des tests manuels avec un simulateur d'application Android pour la conception côté client.

Pour la conception métier, j'ai employé des outils populaires comme Postman et Thunder Client, couramment utilisés par les développeurs pour tester, documenter et interagir avec des API.



cf tests manuel : annexe E.1

Le plan des tests

1. Introduction

Mon objectif est de s'assurer que l'application Tranquillo Organizer fonctionne correctement et répond aux besoins des utilisateurs avec TDAH et TSA mais aussi de valider toutes les fonctionnalités de l'application sur différents scénarios d'utilisation.

2. Stratégie de Test

Les différents types de Tests:

- Tests Unitaires
- Tests fonctionnels
- Tests d'intégration
- Tests de performance
- Tests de sécurité
- Tests d'utilisabilité

- Tests de compatibilité

Outils de Test pour mon projet

- Simulateur d'application Android
- Postman
- PHPUnit (pour tests automatisés)
- JMeter (pour tests de performance)

3. Type de Tests à réaliser

a. Fonctionnels

- Tests de Connexion et Inscription
- Tests de Gestion des Tâches
- Tests de Notifications et Rappels
- Tests de Personnalisation de l'Interface
-

c. Tests de Performance

- Tests de Charge
- Tests de Temps de Réponse
-

e. Tests d'Utilisabilité

- Tests d'Expérience Utilisateur
- Tests de Compatibilité de l'Appareil
-

g. Les tests Unitaires

Les tests unitaires vérifient que chaque petite partie d'un programme fonctionne correctement. Ils aident à détecter les erreurs rapidement lors des modifications du code.

Avec PHP et Symfony, j'utilise le composant PHPUnit pour les tests unitaires.

💡 En préparant un exemple pour ce dossier, j'ai découvert une erreur dans mon code :

Le DTO responsable de la validation du libellé du titre de la tâche refusait les espaces. Cette découverte a été intéressante et utile, car j'ai pu corriger le problème et améliorer la validation des entrées :

```

18
19     public function testSetNameWithValidCharacters()
20     {
21         $taskDto = new TaskDto();
22         $taskDto->setName('Test Task');
23
24         $errors = $this->validator->validate($taskDto);
25
26         $this->assertCount(0, $errors, (string) $errors);
27     }
28

```

b. Tests d'Intégration

- Tests de Connexion API
- Tests de Synchronisation des Données

d. Tests de Sécurité

- Tests d'Authentification et d'Autorisation
- Tests de Vulnérabilité

f. Documentation et Reporting

- Suivi des Tests :
- Rapports de Test

Exemples d'un test manuel sur Thunder

Status: 200 OK Size: 1.28 KB Time: 1.51 s	
Response	Headers 7 Cookies Results 6 Docs
Test Results (6/6)	
Test	Result
Response Code equal to 200	Pass
Response Body is JSON	Pass
Response Body contains "token"	Pass
Header Content-Type equal to application/json	Pass
Response Body contains "code": 202	Pass
Response Body contains "user":	Pass

cf autres tests : annexe E.2

le

Déploiement



1. NativeScript

Pour déployer localement une application NativeScript, j'ai installé NativeScript CLI et les dépendances nécessaires pour Android et iOS.

J'ai créé un nouveau projet avec la commande `tn create` et j'ai exécuté l'application sur un émulateur ou un appareil connecté (application **Native Preview**) en utilisant `tn run`.

NativeScript facilite le développement de véritables applications mobiles natives avec une seule base de code JavaScript ou TypeScript.

Pour la mise en Production d'une Application sur le Google Play Store (Android)

- J'ai créé un compte développeur sur Google Play Console et payer les frais uniques de 25€ ~.
- Il me faut préparer les icônes, captures d'écran, descriptions et autres informations nécessaires.
- Je dois utiliser Android Studio pour générer un APK ou un AAB signé.
- Il me faut soumettre l'application via la Google Play Console en remplissant les détails de distribution.
- Après la révision, qui prend quelques heures à quelques jours, mon application sera disponible sur le Google Play Store.

Mise en Production d'une Application sur l'App Store (Apple)

- Je dois créer un compte développeur sur Apple Developer et payer l'abonnement annuel de 99€ ~.
- Il me faut préparer les icônes, captures d'écran, descriptions et autres informations nécessaires.
- Je dois utiliser Xcode pour générer un fichier IPA signé et le soumettre via App Store Connect.
- Après la révision, qui peut prendre jusqu'à une semaine, mon application sera disponible sur l'App Store.

2. PHP Symfony

Pour le déploiement local d'une application Symfony, j'ai installé PHP et Composer.

Après avoir cloné mon projet Symfony, j'ai utilisé `composer install` pour installer les dépendances.

J'ai configuré mon serveur local avec des commandes comme Symfony `server:start` sans oublier de configurer mon adresse DNS pour pointer vers le répertoire **public/** de mon projet Symfony. Cela sécurise toute la structure Symfony et mes variables d'environnement les rendant inaccessible depuis le web.

Pour la mise en Production de Symfony sur un Serveur Ionus avec Connexion SSH

- Je dois me connecter au serveur via SSH en utilisant un terminal.
- Je transfère les fichiers de mon projet Symfony sur le serveur via le terminal ssh.
- Je navigue dans le répertoire du projet sur le serveur et j'installe les dépendances avec Composer.
- Je configure les variables d'environnement dans le fichier .env.local.
- J'effectue la mise en production de la base de données
- Je fais des essais
- J'active l'API la rendu accessible au utilisateur

3. MariaDB avec Docker

Pour déployer MariaDB localement avec Docker, j'ai installé Docker.

J'ai créé un fichier docker-compose.yml pour définir le service MariaDB avec les configurations nécessaires, puis j'ai lancé le conteneur MariaDB avec la commande **docker-compose up -d**.

J'ai pu accéder à la base de données via un client MySQL en utilisant les informations de connexion définies dans le fichier **compose.yaml**.

Pour la mise en Production de la Base de Données MariaDB sur un Serveur Ionus

- Je me connecte au serveur via SSH en utilisant un terminal.
- Je me connecte à MariaDB en tant qu'utilisateur root.
- Je crée une nouvelle base de données pour mon application.
- Je transfère mon fichier SQL sur le serveur via ma connexion ssh.
- J'importe le fichier SQL dans la nouvelle base de données.
- Je crée le compte Webmaster
- Je poursuis les tests de mon API

Après la formation :

La prochaine étape consistera à déployer mon API sur mon serveur accessible via l'adresse sécurisée '<https://tranquillo.corbisier.fr>'.

La base de données sera également déployée sur le serveur, mais son accès sera restreint : elle ne sera accessible que par l'API et via mon compte utilisateur chez le fournisseur de domaine et de serveur.

La dernière étape consistera à déployer l'application mobile, d'abord sur la plateforme Android, puis sur la plateforme Apple.



Veille Sécurité



➤ API REST

💻 Une API est une interface de programmation d'application (Application Programming Interface) qui permet à un ordinateur de demander une information à un autre ordinateur via internet.

📞 A l'exemple d'un annuaire téléphonique qui permet à un humain de demander une information à un autre humain, par téléphone .

Sur le web, les API jouent le même rôle que les annuaires. Sauf que le téléphone est remplacé par internet et les humains par des ordinateurs.



💻 Une API REST (REpresentational State Transfer) est un type de service web qui utilise un ensemble de contraintes pour permettre aux systèmes de manipuler des ressources web via des requêtes HTTP standard (comme GET, POST, PUT, DELETE) et des formats de données comme HTML, XML, ou JSON. Elle permet une interopérabilité entre ordinateurs sur Internet en offrant une manière uniforme de gérer des ressources identifiées par des URL.

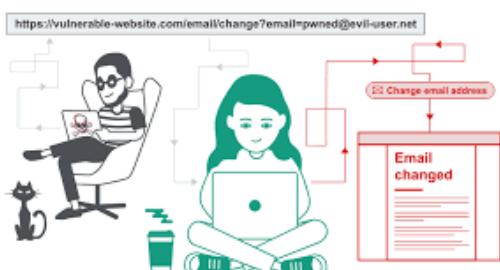
Une de ces particularités est son statut Stateless. Elle ne sauvegarde pas l'état du client côté serveur. On parle d'un "protocole sans état".

Revenons à notre faille de sécurité qui a pour but de tromper le client et interagir avec l'API à son insu.

➤ CSRF

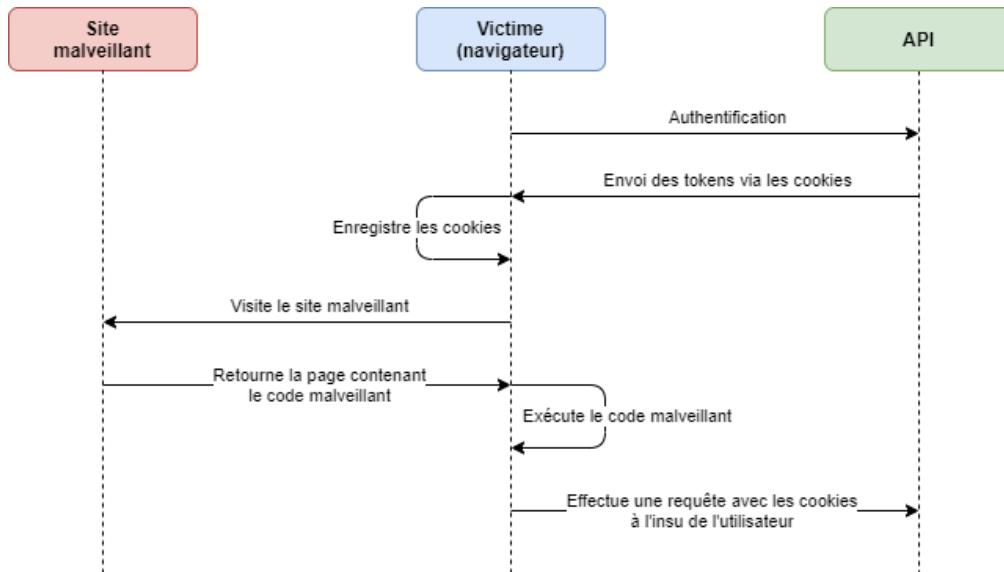
→ Qu'est ce que c'est ?

La faille CSRF (Cross Site Request Forgery) consiste à inciter une victime à exécuter une requête HTTP à son insu. Voici un exemple simple pour illustrer :



1. Un utilisateur est authentifié sur notre API, et le cookie contenant le JWT est stocké dans son navigateur.
2. Une personne mal intentionnée incite l'utilisateur à visiter une page web contenant du code malveillant qui envoie une requête à notre API.
3. La requête est envoyée à notre API avec le cookie JWT de l'utilisateur, automatiquement ajoutée par le navigateur.
4. Notre API exécute l'opération correspondant à la requête, sans que l'utilisateur en soit conscient.

→ Connexion à l'API REST sans JWT

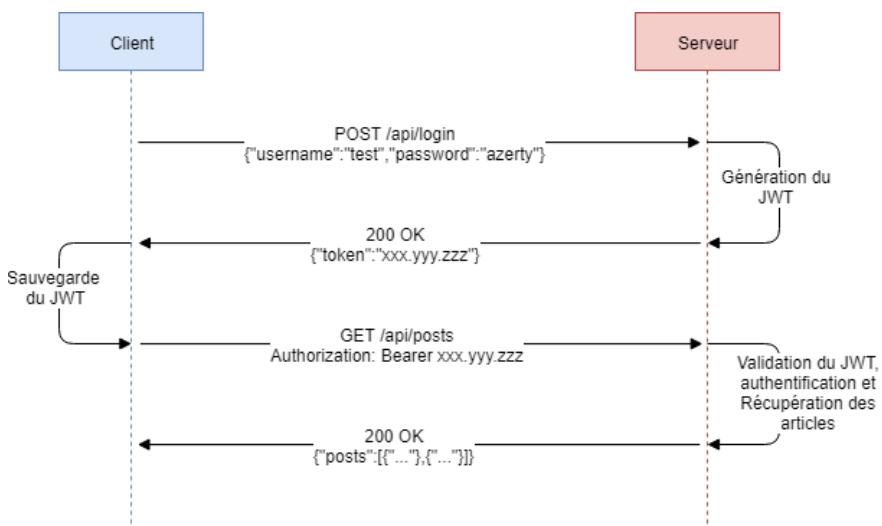


→ Connexion à l'API REST avec JWT

Nous utilisons un token d'authentification de type JWT

🔑 Les JSON Web Tokens (JWT) sont des jetons créés par un serveur lors de l'authentification d'un utilisateur sur une application web, et sont ensuite transmis au client. Ces jetons sont renvoyés avec chaque requête HTTP au serveur, permettant ainsi d'identifier l'utilisateur.

🔑 Les informations contenues dans le jeton sont signées à l'aide d'une clé privée détenue par le serveur. Lors de la réception du jeton, le serveur compare la signature envoyée par le client avec celle qu'il génère à partir de sa propre clé privée. Si les signatures correspondent, le jeton est validé.



Exemple de code si JWT est accepté :

```
18 final class UsersJWTResponsesListener extends AbstractController
19 {
20
21
22     /**
23      * @param AuthenticationSuccessEvent $event
24      */
25     public function onAuthenticationSuccessResponse
26     (AuthenticationSuccessEvent $event)
27     {
28         $data = $event->getData();
29         $user = ObjectHydrator::hydrate($event->getUser(), new UserDto);
30
31         if (!$user instanceof UserDto) {
32             return;
33         }
34
35         $data['user'] = $this->jsonEncode(
36             $user,
37             ['groups' => ['users: read']]
38         );
39         $data['code'] = Response::HTTP_ACCEPTED;
40
41         $event->setData($data);
42     }
43 }
```

Exemple de code si JWT est refusé :

```
48
49
50     /**
51      * @param AuthenticationFailureEvent $event
52      */
53     public function onAuthenticationFailureResponse
54     (AuthenticationFailureEvent $event)
55     {
56         $message = "Incorrect credentials, please check that your username/
57         password is set correctly.";
58
59         $data = [
60             'status' => "Access forbidden",
61             'message' => $message,
62         ];
63
64         $response = new JWTAuthenticationFailureResponse($message,
65             JsonResponse::HTTP_FORBIDDEN);
66         $response->setData($data);
67
68         $event->setResponse($response);
69     }
70
71
72     /**
73      * @param JWTInvalidEvent $event
74      */
75     public function onJWTInvalid(JWTInvalidEvent $event)
76     {
77         $message = "Your token is invalid, please log in again to get a
78         new one.";
79         $data = [
80             'status' => "Access unauthorized",
81             'message' => $message,
82         ];
83
84         $response = new JWTAuthenticationFailureResponse($message,
85             Response::HTTP_UNAUTHORIZED);
86         $response->setData($data);
87
88         $event->setResponse($response);
89     }
90 }
```

→ Problème de sécurité côté client

1. LocalStorage

Les objets de stockage web, localStorage et sessionStorage, permettent d'enregistrer des paires clé/valeur dans le navigateur. Donc, nous pourrions enregistrer le token JWT.

Ce qui est intéressant, c'est que les données stockées dans localStorage persistent même après un redémarrage du navigateur, tandis que celles de sessionStorage survivent uniquement à une actualisation de la page.

!? Pourquoi utiliser ces objets alors que nous avons déjà des cookies ?

Contrairement aux cookies, les objets de stockage web ne sont pas envoyés au serveur à chaque requête, permettant de stocker beaucoup plus de données (au moins 5 mégaoctets dans la plupart des navigateurs). De plus, le serveur ne peut pas manipuler ces objets via les en-têtes HTTP; tout se fait en JavaScript.

Le stockage est lié à l'origine (domaine/protocole/port), ce qui signifie que différents protocoles ou sous-domaines ont leurs propres objets de stockage, et ne peuvent pas accéder aux données des autres.

Ils ne peuvent pas être exploités par une attaque CSRF, puisque inaccessible depuis une autre page du navigateur.

Mais voilà, étant manipulable par du code javascript, cela les rend vulnérables aux attaques XSS, qui consistent à exécuter du code javascript à l'insu du client.

2. Les cookies

Les cookies sont des petits fichiers de données que les sites web stockent dans le navigateur de l'utilisateur.

Ils servent principalement à trois fins :

1. maintenir les sessions de connexion,
2. suivre les activités des utilisateurs sur le site,
3. et stocker des préférences ou des informations spécifiques pour améliorer l'expérience utilisateur.

Contrairement aux objets de stockage web (localStorage et sessionStorage), les cookies sont envoyés au serveur avec chaque requête HTTP, ce qui les rend utiles pour les opérations nécessitant une communication serveur-client régulière, comme l'authentification persistante.

Les cookies ont une taille limitée (généralement environ 4 Ko par cookie) et peuvent être configurés avec des attributs spécifiques, tels que la date d'expiration, le domaine, le chemin, et les indicateurs de sécurité (Secure et HttpOnly).

- **Expiration :**

Par défaut, les cookies sont supprimés à la fermeture du navigateur (cookies de session). Toutefois, ils peuvent être configurés pour expirer à une date précise ou après une certaine durée (cookies persistants).

- **Domaine et chemin :**

Les cookies peuvent être limités à un domaine et un chemin spécifique, contrôlant ainsi leur portée d'accès.

- **Sécurité :**

Les attributs Secure et HttpOnly améliorent la sécurité en assurant que les cookies sont transmis uniquement via des connexions sécurisées (HTTPS) et en rendant les cookies inaccessibles via JavaScript, cela les rend vulnérables aux attaques XSS.

Les cookies sont des outils essentiels pour la gestion des sessions et le suivi des utilisateurs, mais cela les rend également vulnérables aux attaques CSRF.

→ Quelle peut être la solution ?



Et pourquoi ne pas utiliser les deux, le localStorage ET le cookie.

Pour cela, nous allons mettre à contribution notre serveur API

→ nous allons lui demander de générer un token CSRF :

Voici les étapes :

→ **Suite à la demande de connexion, réussi, du client :**

Côté serveur :

- Générer un token CSRF
- Générer un token JWT
- Stocker le token CSRF dans le payload du token JWT
- Envoyer le jwt dans un cookie
- Envoyer le token CSRF dans une entête HTTP `x-xsrf-token`



Côté client :

- Enregistrement du cookie avec les sécurités adaptées
- Enregistrement du token CSRF dans le local storage

→ **Lors d'une requête :**

Côté client :

- Envoie du cookie avec les sécurités adaptées dans la requête
- Envoie du token CSRF dans le header de la requête
- Envoie du body et/ou de la route

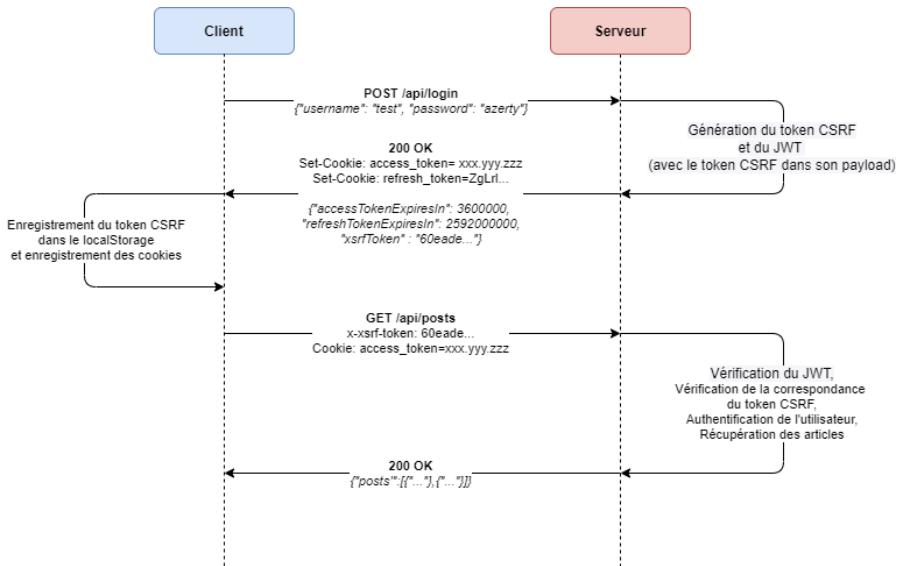


Côté serveur :

- Décoder le token JWT et vérification de sa validité
- Récupérer le token CSRF et vérifie s'il est identique au token envoyé dans le header

Si la réponse est valide :

- Envoie de la réponse au client avec
 - un nouveau token JWT
 - un nouveau token CSRF
 - le contenu de la requête et/ou la réponse de la requête



➤ Pour résumer

Il existe plusieurs solutions pour améliorer la sécurité des API web, et l'une d'entre elles consiste à combiner localStorage et cookies pour la gestion des tokens JWT et CSRF. En utilisant cette méthode :

1. Stockage des Tokens :

- **JWT (JSON Web Token) :** Le token JWT est stocké dans un cookie sécurisé avec les attributs HttpOnly et Secure. Cela empêche l'accès au token par du code JavaScript malveillant, réduisant ainsi le risque d'attaques XSS.
- **Token CSRF :** Le token CSRF est stocké dans le localStorage. Bien que cela le rende vulnérable aux attaques XSS, il est protégé contre les attaques CSRF car il n'est pas automatiquement envoyé avec chaque requête HTTP par le navigateur.

2. Validation des Requêtes :

- À chaque requête, le client envoie le cookie contenant le JWT ainsi que le token CSRF dans un en-tête HTTP personnalisé (par exemple, `x-xsrf-token`).
- Le serveur décode le JWT pour vérifier son intégrité et extrait le token CSRF du payload.
- Le serveur compare ensuite le token CSRF envoyé dans l'en-tête avec celui extrait du JWT. Si les deux correspondent, la requête est validée.

3. Renouvellement des Tokens :

- Pour chaque réponse, le serveur peut envoyer un nouveau token JWT et un nouveau token CSRF, garantissant ainsi que les tokens sont régulièrement mis à jour et que les sessions restent sécurisées.

Cette approche utilise les avantages de localStorage et des cookies tout en atténuant leurs faiblesses respectives, offrant une solution robuste contre les vulnérabilités courantes telles que les attaques CSRF et XSS. C'est une des nombreuses stratégies disponibles pour sécuriser les applications web.

Résumé du Projet Tranquillo Organizer

L'application Tranquillo Organizer est conçue pour aider les personnes ayant entre autres un Trouble Déficitaire de l'Attention avec Hyperactivité (TDAH) ou un Trouble du Spectre Autistique (TSA) à mieux organiser leur quotidien. Elle offre des fonctionnalités comme la création de rappels flexibles, la visualisation des délais, des options de personnalisation comme le mode nuit, et un journal de suivi.

L'application est spécialement destinée aux personnes ayant des besoins particuliers en matière de gestion du temps et de l'organisation, notamment les personnes avec TDAH et TSA.

Pour le développement de l'API, j'ai utilisé PHP avec le framework Symfony, qui s'appuie sur l'ORM Doctrine pour la gestion de la base de données. La base de données est MariaDB, déployée localement à l'aide de Docker.

Pour l'interface utilisateur, j'ai développé avec NativeScript qui facilite le développement de véritables applications mobiles natives en utilisant une seule base de code JavaScript ou TypeScript.

En complément, j'ai utilisé Svelte Native pour créer une application mobile performante et réactive, en tirant parti d'un code JavaScript optimisé pour interagir directement avec les composants natifs.

Le déploiement de l'API se fera sur un serveur via une connexion sécurisée SSH, et l'API sera accessible à l'adresse sécurisée '<https://tranquillo.corbisier.fr>'. La base de données ne sera accessible que par l'API et mon compte utilisateur chez le fournisseur de domaine et de serveur.

L'application mobile sera déployée sur les plateformes Android et Apple.

L'application est structurée en trois couches principales :

1. les contrôleurs gèrent les requêtes HTTP et orchestrent les appels aux services métiers ;
2. les services contiennent la logique métier et les opérations complexes ;
3. les repositories gèrent l'accès aux données via Doctrine ORM.

La sécurité est primordiale, avec l'utilisation de JWT pour l'authentification, des tokens de rafraîchissement, et des mécanismes pour prévenir les attaques CSRF et XSS. Le hashage des mots de passe assure leur protection.

Un plan de tests détaillé a été élaboré pour vérifier l'inscription, la connexion, la gestion des tâches, les notifications, la performance, et la sécurité. Les tests garantissent que l'application fonctionne correctement et répond aux besoins des utilisateurs.

Tranquillo Organizer est une application mobile destinée à être robuste et sécurisée, conçue pour aider les personnes avec TDAH et TSA à organiser leur quotidien. Le développement, le déploiement et les tests assurent que l'application offre une expérience utilisateur fluide et sécurisée.

ANNEXES

A. Projet

A.1. User storie

★ User Story 3 :

- **En tant qu'** enseignant travaillant avec des enfants atteints de TDAH,
- **Je veux** un calendrier facile à utiliser pour planifier mes journées de travail et mes rendez-vous personnels,
- **Afin de** maintenir un équilibre entre mon travail et ma vie personnelle.

★ User Story 4 :

- **En tant qu'** utilisateur avec des besoins de gestion de tâches spécifiques,
- **Je veux** des alertes visuelles et sonores variées,
- **Afin de** recevoir des notifications de manière personnalisée, répondant à mes préférences.

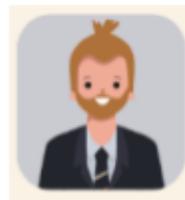
★ User Story 5 :

- **En tant qu'** utilisateur,
- **Je veux** mettre en pause mes rappels lorsque nécessaire,
- **Afin de** pouvoir me concentrer sur d'autres tâches urgentes et reprendre mes rappels plus tard.

★ User Story 6 :

- **En tant qu'** utilisateur de l'application,
- **Je veux** lier mes rappels à ma routine quotidienne,
- **Afin de** créer des alertes spécifiques pour les moments clés de ma journée.

A.2. Personas



MARC

Âge
entre 25 et 30 ans

Profession
Développeur informatique

Personnalité
Méticuleux et analytique
logique et précision
réservé et préfère souvent
travailler en solitaire.

Centres d'intérêts
programmation
nouvelles technologies
jeux vidéo
science-fiction.

Problèmes

développeur informatique talentueux travaillant dans une entreprise de technologie, Marc a un TSA. Il peut souvent rencontrer des difficultés dans la gestion du temps et des tâches.

Défis

"J'aime quand tout est bien organisé et structuré. Cela me permet de me concentrer sur ce que je fais et d'être plus productif dans mon travail de développement."

Frustrations

se sent parfois dépassé par la gestion de multiples tâches et peut avoir du mal à communiquer efficacement avec les autres membres de l'équipe

Besoins

- ⌚ avoir une application visuelle et intuitive pour organiser ses projets de programmation et suivre leur progression de manière efficace
- ⌚ fonctionnalités de rappel pour l'aider à rester concentré sur ses tâches et à éviter les distractions
- ⌚ options de personnalisation pour adapter l'application à ses préférences et à ses besoins spécifiques en matière de planification
- ⌚ outils de gestion du temps pour estimer le temps nécessaire à l'accomplissement de chaque tâche et éviter le surmenage.



SOPHIE

Âge
entre 40 et 45 ans

Profession
Professeur des écoles

Personnalité
organisée, empathique
cherche constamment des
solutions pratiques pour
faciliter son quotidien

Centres d'intérêts
la lecture
la randonnée en plein air dans
la nature pour se ressourcer
la cuisine et prendre plaisir à
découvrir de nouvelles recettes

Problèmes

Enseignante passionnée qui travaille avec des enfants dont certains sont atteints de TDAH (Trouble Déficitaire de l'Attention avec ou sans Hyperactivité), elle jongle avec de nombreuses tâches au quotidien, y compris la préparation des cours, la gestion de la classe et le suivi des progrès des élèves. Malgré son emploi du temps chargé, elle essaie de maintenir un équilibre entre son travail et sa vie personnelle.

Défis

"J'apprécie la satisfaction de voir mes élèves progresser et réussir. L'organisation est essentielle pour moi, car elle me permet de maintenir un équilibre entre ma vie professionnelle et personnelle."

Frustrations

se sentir dépassée par la charge de travail et avoir du mal à se rappeler de toutes les tâches à accomplir et lorsque son emploi du temps ne lui permet pas de consacrer du temps à sa vie personnelle et à ses loisirs.

Besoins

- ⌚ avoir un calendrier facile à utiliser pour planifier ses journées de travail et ses rendez-vous personnels
- ⌚ avoir des rappels pour ne pas oublier les réunions d'équipe, les séances de suivi individuel avec les élèves et les rendez-vous médicaux
- ⌚ avoir des fonctionnalités de liste de tâches pour prioriser les activités et rester organisée tout au long de la journée et ceux, selon l'endroit où elle se trouve.
- ⌚ un outils de suivi pour évaluer sa productivité et identifier les moments où elle est la plus efficace.

B. Kanban

B.1. Liste issues (workflow)

The screenshot shows the Tranquillo Organizer interface for the 'ecorbisierSimplon / tranquillo' repository. The 'Issues' tab is selected, displaying 9 open issues. The issues listed are:

- FRONT Login : Créer les tests unitaires pour le login (#94)
- Front Login : créer les tests unitaires pour la connexion (#93)
- FRONT Register : Créer les test fonctionnels (#92)
- FRONT Registers : Creer les tests (#91)
- Back test (#90)
- Back Test Update (#89)
- Ajout htmlspecialchars au front pour la description (#87)
- BACK - Refactorise les services (#74)
- BACK Login : Créer les tests unitaires pour le login (#46)

Filters: is:issue is:open

Author, Label, Projects, Milestones, Assignee, Sort dropdowns.

New issue button.

B.2. Exemple Issues (ou tickets)

The screenshot shows the details of issue #78, titled "FRONT Ajout du module d'inscription".

Issue Details:

- Status: Closed
- Assignee: ecorbisierSimplon
- Labels: None
- Projects: tranquillo
- Priority: Priority
- Size: Size
- Estimate: Enter number...
- Start date: No date
- End date: No date

Description:

Création du module d'inscription.
Ce module contiendra :

Nom du label	Nom technique
Nom	lastname
Prenom	firstname
Email	email
Mot de passe	password
Confirmation du mot de passe	password_repeat

Comments:

- ecorbisierSimplon self-assigned this 2 weeks ago
- ecorbisierSimplon added this to tranquillo 2 weeks ago
- ecorbisierSimplon mentioned this 2 weeks ago
- Eric/2024_05_08-01:45:38/Ajout du module d'inscription #79
- ecorbisierSimplon closed this as completed in #79 2 weeks ago

B.3. le tableau Kanban

un tableau Kanban ; les deux images suivantes représentent un tableau dans sa continuité :

Type to search Add stat

My items + New view

● In progress 2 / 3 Estimate: 0 ...

This is actively being worked on

- tranquillo #87 Ajout htmlspecialchars au front pour la description
- tranquillo #93 Front Login : créer les tests unitaires pour la connexion

○ In review 0 / 5 Estimate: 0 ...

This item is in review

○ In test 0 Estimate: 0 ...

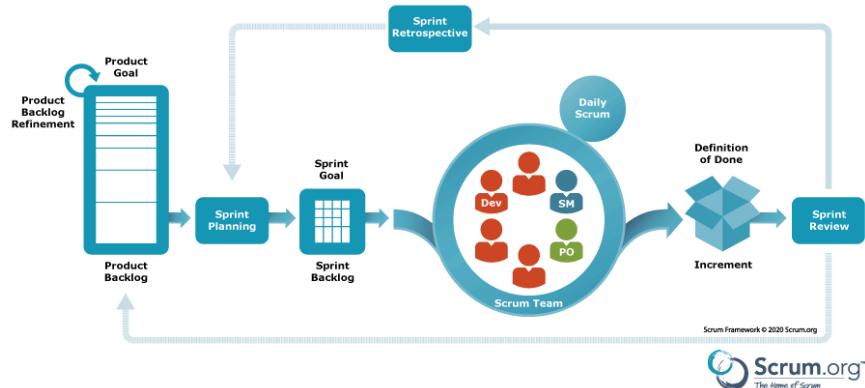
● Done 32 Estimate: 0 ...

This has been completed

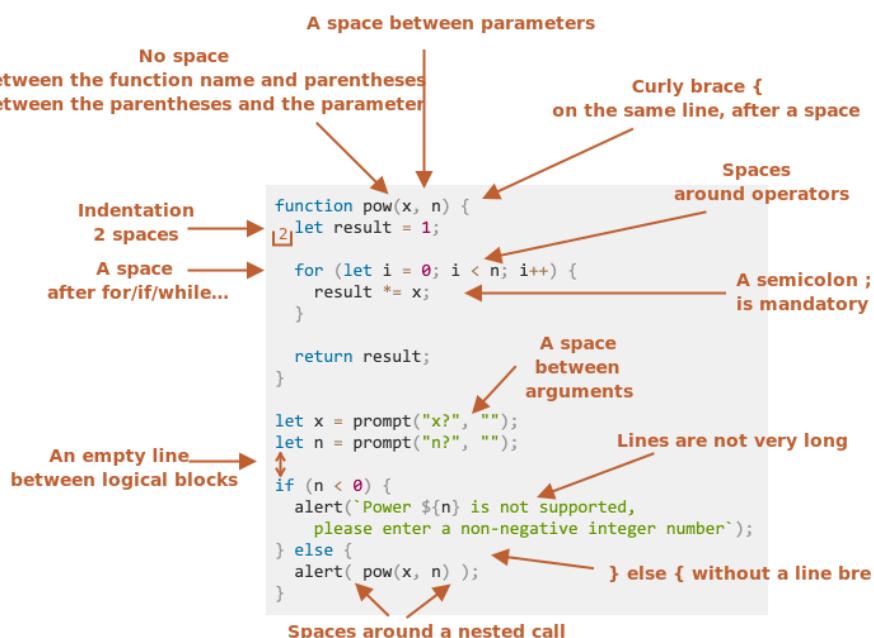
- tranquillo #70 Ajout des vérifications des droits users
- tranquillo #72 FRONT - Ajout du module login
- tranquillo #76 "FRONT - Terminer le login"
- tranquillo #78 FRONT Ajout du module d'inscription
- tranquillo #80 Modification du visuel du menu

B.4. Cadre de travail SCRUM

SCRUM FRAMEWORK



B.5. Exemple de bonne pratiques



C. La conception

C.1. Svelte Native

C.1.1. Structure

```
✓ tranquilloApp
  > .vscode
  ✓ app
    ✓ components
      > layout
    ✓ task
      ⚡ BackButton.svelte
      ⚡ ChoiceTime.svelte
      ⚡ CreateTask.svelte
      TS ListReminder.ts
      ⚡ Task.svelte
      ⚡ TaskList.svelte
    ✓ user
      ⚡ Login.svelte
      ⚡ Register.svelte
      ⚡ forms.svelte
      ⚡ Home.svelte
    > css
```

```
✓ tranquilloApp
  ✓ app
    ✓ components
      ✓ user
        ⚡ forms.svelte
        ⚡ Home.svelte
    > css
    > fonts
    ✓ i18n
      TS en_US.ts
      TS fr_FR.ts
    > lib
    ✓ models
      TS profile.ts
      TS task.ts
      TS user.ts
    ✓ scss
      ⚡ _buttons.scss
      ⚡ _fonts.scss
    > stores
    > utils
      ⚡ app.scss
      ⚡ App.svelte
    TS app.ts
    > App_Resources
    > hooks
```

C.1.2. Code en Svelte

triquilloApp > app > components > task > Task.svelte > script

```

1<script lang="ts">
2  async function onTapNavValidate(): Promise<void> {
3    if (disabled === "") {
4      isLoading = true;
5      disabled = "disabled";
6
7      startAt = startAt ? dtDisplay.datetime(startAt) : null;
8      endAt = endAt ? dtDisplay.datetime(endAt) : null;
9
10     const res: ErrRespTask = await control.task({
11       name,
12       description,
13       reminder,
14       startAt,
15       endAt,
16     });
17
18     if (res.ok) {
19       const id: number = task.id ?? 0;
20       taskStore
21         .updateTask(
22           { id, name, description, reminder, startAt, endAt },
23           userToken,
24         )
25         .then(
26           () => {
27             goBack();
28           },
29           async (err) => {
30             console.log(err.errors);
31             if (err.errorCode === 422) {
32               if (!err.errors || !err.errors.errors) {
33                 alert({
34                   title: localize("message.title.HTTP_500"),
35                   message: localize("message.error.not_registered", true),
36                 });
37               } else {
38                 let msg = "";
39                 let errs = err.errors.errors;
40                 for (let field of Object.keys(errs)) {
41                   msg += `${field}: ${errs[field][0]}\n`;
42                 }
43                 alert({
44                   title: localize("message.title.err_validation"),
45                   message: msg,
46                 });
47               }
48             } else {
49               const codeHttp: number = err.errors.status;
50               if (codeHttp === 409) {
51                 // errMessage.email = "this email has exited";
52               } else {
53                 alert({
54                   title: localize("message.title.HTTP_" + codeHttp),
55                   message: localize("message.error.http_error", true),
56                 });
57               }
58             }
59           );
60           }
61         );
62       }
63     );
64   }
65
66   if (res.ok) {
67     const id: number = task.id ?? 0;
68     taskStore
69       .updateTask(
70         { id, name, description, reminder, startAt, endAt },
71         userToken,
72       )
73       .then(
74         () => {
75           goBack();
76         },
77         async (err) => {
78           console.log(err.errors);
79           if (err.errorCode === 422) {
80             if (!err.errors || !err.errors.errors) {
81               alert({
82                 title: localize("message.title.HTTP_500"),
83                 message: localize("message.error.not_registered", true),
84               });
85             } else {
86               let msg = "";
87               let errs = err.errors.errors;
88               for (let field of Object.keys(errs)) {
89                 msg += `${field}: ${errs[field][0]}\n`;
90               }
91               alert({
92                 title: localize("message.title.err_validation"),
93                 message: msg,
94               });
95             }
96           } else {
97             const codeHttp: number = err.errors.status;
98             if (codeHttp === 409) {
99               // errMessage.email = "this email has exited";
100             } else {
101               alert({
102                 title: localize("message.title.HTTP_" + codeHttp),
103                 message: localize("message.error.http_error", true),
104               });
105             }
106           );
107         }
108       );
109     }
110   }
111   isLoading = false;
112   disabled = "";
113 }
114
115 } else {
116   const codeHttp: number = err.errors.status;
117   if (codeHttp === 409) {
118     // errMessage.email = "this email has exited";
119   } else {
120     alert({
121       title: localize("message.title.HTTP_" + codeHttp),
122       message: err.errors.detail,
123     });
124   }
125   isLoading = false;
126   disabled = "";
127 }
128
129 function onTapNavCancel(): void {
130   goBack();
131 }
132
133 function onTapDelete(): void {
134   isLoading = true;
135   disabled = "disabled";
136   confirm({
137     title: localize("message.title.delete_task", true),
138     message: printt(localize("message.confirm_delete"),
139       task.name),
140     okButtonText: localize("yes", true),
141     cancelButtonText: localize("no", true),
142   }).then(async (res) => {
143     if (res) {
144       let result: number =
145         (await taskStore.deleteTask(task.id || 0,
146           userToken)) ?? 0;
147       goBack();
148     }
149   });
150   isLoading = false;
151   disabled = "";
152 }
153
154 </script>
```

triquilloApp > app > components > task > Task.svelte > page > actionBar > stackLayout > BackButton

```

146 <page>
147   <actionBar title="">
148     <stackLayout orientation="horizontal" horizontalAlignment="left">
149       <BackButton />
150       <stackLayout orientation="vertical">
151         <label text={task_name} class="author-name" />
152         <label text={task_date} class="date" />
153       </stackLayout>
154     </stackLayout>
155   </actionBar>
156
157 <scrollView>
158   <stackLayout class="task-content">
159     <stackLayout class="form">
160       <stackLayout orientation="horizontal">
161         <label
162           class="label-icon icon calendar-alt"
163           text={icons.calendar_alt}
164         />
165         <textView
166           bind:this={name_edit}
167           on:textChange={(event) =>
168             name = event.value;
169           }
170           hint={localize("task.name")}
171           text={task.name}
172           class="task-name input"
173           autocapitalizationType="none"
174           returnKeyType="next"
175           editable={!isLoading}
176         />
177       </stackLayout>
178     </stackLayout>
179     {#if errMessage?.name}
180       <stackLayout class="error">
181         <label text={errMessage?.name} />
182       </stackLayout>
183     {/#if}
184     <stackLayout orientation="horizontal">
185       <label
186         class="label-icon icon format-align-left"
187         text={icons.format_align_left}
188       />
189       <textView
190         bind:this={description_edit}
191         on:textChange={(event) =>
192           description = event.value;
193           }
194           hint={localize("task.description")}
195           text={task.description}
196           class="task-body input"
197           autocapitalizationType="none"
198           returnKeyType="next"
199           editable={!isLoading}
200         />
201     </stackLayout>
202   </stackLayout>
203 </page>
```

triquilloApp > app > components > task > Task.svelte > page > actionBar > stackLayout > BackButton > style

```

146 <page>
147   <actionBar title="">
148     <stackLayout>
149       <scrollView>
150         <stackLayout class="task-content">
151           <stackLayout class="hr-light" />
152           <activityIndicator
153             busy={isLoading}
154             horizontalAlignment="center"
155             verticalAlignment="middle"
156             class="activity-indicator"
157           />
158         </stackLayout>
159       </scrollView>
160     </stackLayout>
161   </actionBar>
162
163 <style lang="scss">
164   datepickerFields {
165     font-size: 15;
166     text-align: center;
167     width: 250;
168   }
169
170   .label {
171     &::icon {
172       width: 40;
173       text-align: center;
174     }
175     &.date {
176       font-size: 10;
177     }
178     &.input {
179       white-space: nowrap;
180       width: 100%;
181     }
182     &.task {
183       &::content {
184         padding: 20 10;
185       }
186       &::name {
187         font-size: 20;
188         color: black;
189         font-weight: bold;
190         margin-bottom: 20;
191       }
192       &::body {
193         font-size: 15;
194         margin-bottom: 20;
195       }
196     }
197     &.hr {
198       background-color: hsla(45, 67%, 71%, 0.582);
199       &::light {
200         margin: 20 0 10 0;
201       }
202     }
203   </style>
```



C.1.3. Utilisation du token JWT

```

ts user.ts ... ts user.ts ts task.ts M x
tranquilloApp > app > stores > TS user.ts > ...
58 * qui se résout en un objet utilisateur.
59 * @param {string} email - Adresse e-mail de
l'utilisateur essayant de se connecter.
60 * @param {string} password - Le paramètre « password »
dans la fonction « login » est une chaîne qui
représente le mot de passe de l'utilisateur. Il est
utilisé à des fins d'authentification pour
vérifier l'identité de l'utilisateur lors de la
connexion au système.
61 */
62 export function login(email: string, password: string)
: Promise<User> {
63   return client
64     .sendRequest<UserResponse>("/login_check", "POST",
65       null,
66       {
67         username: email.trim(),
68         password: password.trim(),
69       })
70     .then((userResponse) => {
71       // console.log("UserResponse : " + userResponse.
72       token);
73
74       let user: User = userResponse.user as User;
75       user_token.set(userResponse.token as string);
76       user_profile.set(user as User);
77       return user;
78     });
79
80 /**
81 * Cette fonction prend un objet ProfileUpdate en
entrée et renvoie une promesse qui se résout en un
* objet User après la mise à jour du profil.
82 * @param {ProfileUpdate} update - Objet ProfileUpdate
contenant les champs à mettre à jour pour le
* profil d'un utilisateur.
83 */
84 export function update(update: ProfileUpdate):
Promise<User> {
85   let payload: any = {
86     firstname: update.firstname.trim(),
87     lastname: update.lastname.trim()
88   }
89 }

```

```

8 class TaskStore {
42   loadNextPage() {
43   }
44
45   private async createUpdate(
46     task: Task,
47     user_token: string,
48     method: string = "post"
49   ) {
50     let url = "/task/";
51
52     let response = await client.sendRequest<ErrorResponse>(
53       url,
54       method,
55       user_token,
56       task
57     );
58     this.user_token = user_token;
59
60     return response.status;
61   }
62
63   updateTask(task: Task, user_token: string = "") {
64     return this.createUpdate(task, user_token, "put");
65   }
66   createTask(task: Task, user_token: string = "") {
67     return this.createUpdate(task, user_token, "post");
68   }
69
70   private async delete(id: number, user_token: string) {
71     let url = "/task/" + id;
72     // url += `&offset=${page * TASKS_PER_PAGE}&limit=${
73     // TASKS_PER_PAGE}`;
74     let response = await client.sendRequest<ErrorResponse>(
75       url,
76       "delete",
77       user_token
78     );
79     this.user_token = user_token;
80
81     return response.status;
82   }
83 }

```

C.1.4. Code avec Typescript

```

ts LOGIN ... ts LOGIN Erroneous ts Login.svelte ts user.ts ts Task.svelte 2 ts client.ts ...
tranquilloApp > app > lib > TS clients > ApiClient > sendRequest
1 import { ErrorResponse } from "./models/profile";
2 import { EventEmitter } from "./utils/eventemitter";
3
4 // const API_BASE = "http://192.168.1.160:8088/api";
5 const API_BASE = "http://192.168.1.27:8088/api";
6
7 // type ValidationErrors = { [index: string]: string[] };
8
9 class ApiError {
10   constructor(message: string, errorCode: number = 0) {
11     this.message = message;
12     this.errorCode = errorCode;
13   }
14   errors: ErrorResponse = {};
15   errorCode: number = 0;
16   message: string = "Api Error";
17 }
18
19 class ApiClient {
20   onError: EventEmitter<ApiError>;
21
22   constructor() {
23     this.onError = new EventEmitter<ApiError>();
24   }
25
26   async sendRequest<T>(
27     relative_url: string,
28     method: string,
29     token: string | null = null,
30     payload?: {} | null
31   ): Promise<T> {
32     /* Cette instruction « console.log » enregistre un message sur la cons
la requête HTTP en cours. Il concatène la constante 'API BASE' avec le p
pour former l'URL complète demandée. De plus, il inclut la méthode HTTP
33 */
34
35     /* Ce bloc de code configure les en-têtes d'une requête HTTP : */
36     let headers: HeadersInit = {};
37     // if (payload) {
38     //   headers["Content-Type"] = "application/json";
39     //   headers["Accept"] = "application/json";
40     //   headers["Authorization"] = `Bearer ${token}`;
41     // }
42     if (token) {
43       headers["Authorization"] = `Bearer ${token}`;
44     }
45
46     /* Cet extrait de code effectue une requête HTTP asynchrone à l'aide d
JavaScript/TypeScript : */
47     try {
48       let res;
49       try {
50         res = await fetch(`${API_BASE}${relative_url}`, {
51           method: method,
52           mode: "cors",
53           headers: headers,
54           body: payload ? JSON.stringify(payload) : null,
55         });
56       } catch (e) {
57         console.log("error running fetch", e);
58         throw e;
59       }
60
61       /* Ce bloc de code gère les réponses d'erreur de la requête API : */
62       if (!res.ok) {
63         let err = new ApiError(res.statusText, res.status);
64         try {
65           let validation_errors: ErrorResponse = await res.json();
66           console.log(`Error serveur : ${validation_errors.title}`);
67           console.log(`Error serveur : ${validation_errors.status}`);
68           err.errors = validation_errors;
69           console.log(`Error serveur2 : ${err.errors.detail}`);
70         } catch {}
71
72         this.onError.fire(err);
73         throw err;
74       }
75
76       /* Ce bloc de code tente d'analyser la réponse du serveur au format JS
Si l'analyse réussit, elle renvoie les données JSON analysées. Cependant
produit pendant le processus d'analyse (par exemple, si la réponse n'a pas
détecté l'erreur, crée une nouvelle instance 'ApiError' avec le message
l'analyse de la réponse du serveur)., déclenche le 'onError' événement
renvoie l'erreur à gérer par l'appelant de la méthode 'sendRequest'. Ces
problèmes liés à l'analyse de la réponse du serveur sont correctement
l'événement « onError ». */
77       try {
78         return await res.json();
79       } catch {
80         let err = new ApiError("Error parsing server response");
81         this.onError.fire(err);
82         throw err;
83       }
84     }
85   }
86
87   export let client = new ApiClient();
88
89 
```

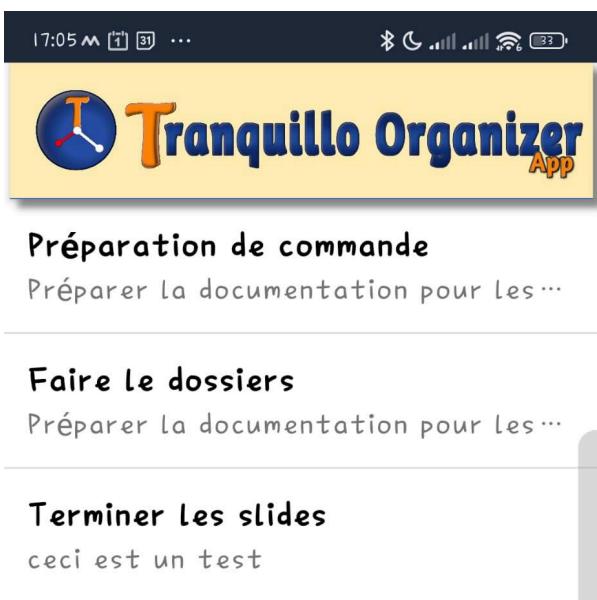
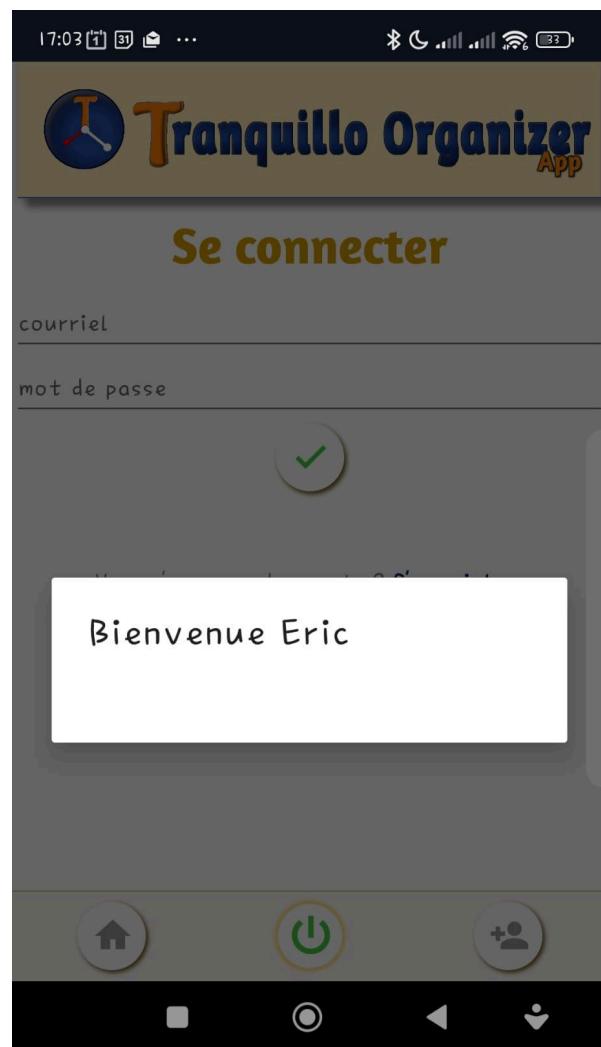
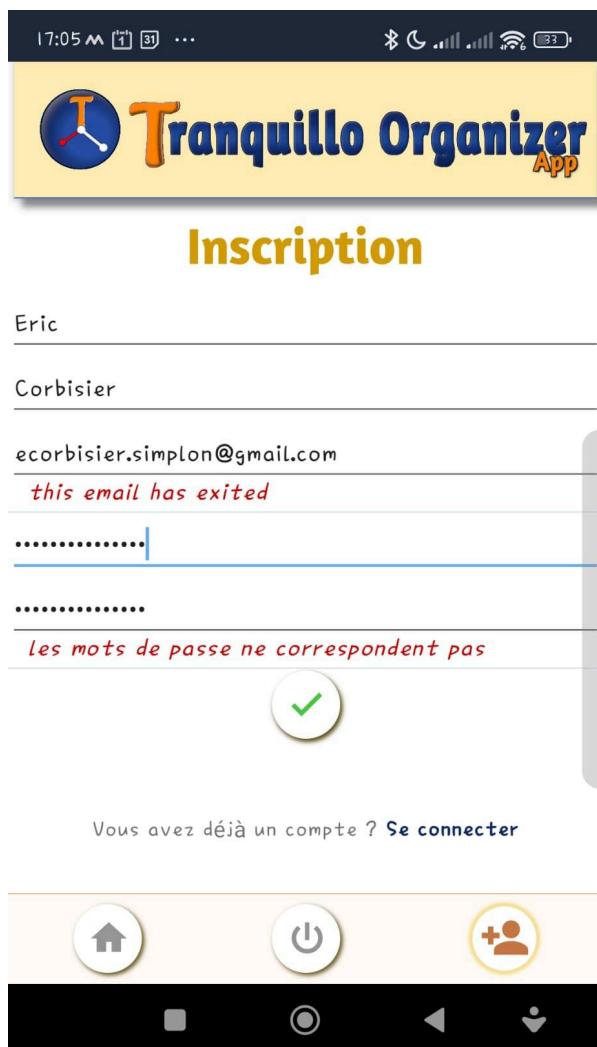
```

ts fr_FR.ts
tranquilloApp > app > i18n > ts fr_FR.ts > ...
1 export const i18n = {
2   name_app: "Tranquillo",
3   yes: "oui",
4   no: "non",
5   date: "la date",
6   time: "l'heure",
7   minutes: "minutes",
8   hours: "heures",
9   days: "jours",
10  previously: "auparavant",
11  none: "aucun",
12  description: "description",
13
14  user: {
15    email: "courriel",
16    password: "mot de passe",
17    password_repeat: "recopier le mot de passe",
18    lastname: "nom",
19    firstname: "prénom",
20  },
21  task: {
22    title: "listes des tâches",
23    no_task: "aucune tâche n'est ici... pour le moment",
24    name: "titre",
25    start_date: "date de début",
26    start_time: "heure de début",
27    end_date: "date de fin",
28    end_time: "heure de fin",
29    start: "à partir de",
30    before: "avant le",
31    delay: "délai d'exécution",
32  },
33  form: {
34    no_account: "vous n'avez pas de compte ?",
35    as_account: "vous avez déjà un compte ?",
36    registration: "inscription",
37    register: "s'enregistrer",
38    login: "se connecter",
39  },
40  dialog: {
41    approve: "approuver",
42    reject: "rejeter",
43    confirm_select: "confirmer $1",
44  },
45  button: {
46    validate: "valider",
47    cancel: "annuler",
48    edit: "éditer",
49    delete: "supprimer",
50    share: "partager",
51  },
52  message: {
53    welcome: "bienvenue",
54    ok_register: "vous pouvez vous connecter",
55    confirm_delete: "voulez-vous vraiment supprimer « $1 » ?",
56    title: {
57      ok_register: "inscription réussie",
58      err_validation: "problème de validation",
59    }
60  }
61
62  messages: {
63    title: {
64      http_100: "Continuer",
65      HTTP_101: "Changement de protocole",
66      HTTP_200: "OK",
67      HTTP_201: "Créé",
68      HTTP_202: "Accepté",
69      HTTP_203: "Informations non faisant autorité",
70      HTTP_204: "Aucun contenu",
71      HTTP_205: "Réinitialiser le contenu",
72      HTTP_206: "Contenu partiel",
73      HTTP_300: "Choix multiples",
74      HTTP_301: "Déplacé définitivement",
75      HTTP_302: "Trouvé",
76      HTTP_303: "Voir Autre",
77      HTTP_304: "Non modifié",
78      HTTP_305: "Utiliser un proxy",
79      HTTP_306: "(Inutilisé)",
80      HTTP_307: "Redirection temporaire",
81      HTTP_400: "Requête incorrecte",
82      HTTP_401: "Non autorisé",
83      HTTP_402: "Piètement requis",
84      HTTP_403: "Accès Interdit",
85      HTTP_404: "Introuvable",
86      HTTP_405: "Méthode non autorisée",
87      HTTP_406: "Non acceptable",
88      HTTP_407: "Authentification proxy requise",
89      HTTP_408: "Délai d'expiration de la demande",
90      HTTP_409: "Conflit",
91      HTTP_410: "Parti",
92      HTTP_411: "Longueur requise",
93      HTTP_412: "Échec de la précondition",
94      HTTP_413: "Entité de demande trop grande",
95      HTTP_414: "URI de requête trop long",
96      HTTP_415: "Type de média non pris en charge",
97      HTTP_416: "Plage demandée non satisfaisante",
98      HTTP_417: "Échec de l'attente",
99      HTTP_422: "Entité non traitable",
100     HTTP_500: "Erreur interne du serveur",
101     HTTP_501: "Non implémenté",
102     HTTP_502: "Mauvaise passerelle",
103     HTTP_503: "Service indisponible",
104     HTTP_504: "Délai d'expiration de la passerelle",
105     HTTP_505: "Version HTTP non prise en charge",
106   },
107   error: {
108     not_logged: "nom d'utilisateur ou/et mot de passe invalide",
109     not_registered: "nom d'utilisateur, e-mail ou mot de passe invalide",
110     email_exist: "cet adresse email existe déjà",
111   },
112   pattern: {
113     name: "Veuillez entrer un(e) '$1' valide !",
114     email: "veuillez entrer une adresse email valide !",
115     password: {
116       length: "le mot de passe doit comporter au moins 8 caractères !",
117       force: "le mot de passe doit contenir au moins une majuscule une minuscule"
118     }
119   }
120 }

```



C.2. Visuel réel sur mobile Android



C.3. Couche métier

C.3.1. Architecture

The screenshot shows the Tranquillo IDE interface. On the left, the 'EXPLORATEUR' sidebar displays the project structure under 'TRANQUILLO'. It includes 'serveur-backend', 'public', 'src' (with subfolders like ApiResource, Controller, DataFixtures, Dto, Entity, EventListener, Helper, Repository, Security, Service, Validator, Kernel.php, templates, and tests), and files like bootstrap.php, EntityPostTest.php, translations, var, vendor, .editorconfig, .env, and .env.local. The main workspace shows the code for 'index.php' in a code editor window. The code starts with the standard PHP opening tag and imports the App\Kernel class. It then defines a function to create a new Kernel instance based on the APP_ENV and APP_DEBUG environment variables.

```
<?php
use App\Kernel;
require_once dirname(__DIR__).'/vendor/autoload_runtime.php';
return function (array $context) {
    return new Kernel($context['APP_ENV'], (bool) $context['APP_DEBUG']);
};
```

C.3.2. Code

C.3.2.1. JWT

The screenshot shows two code editors for 'UsersJWTCreatedListener.php' in the 'EventListener' directory. Both editors show the same class definition. The class extends AbstractController and implements the onJWTCreated event. The code handles the creation of a JWT payload by extracting client information from the request headers and setting expiration and user agent details. The right-hand editor has some code highlighted in yellow.

```
final class UsersJWTCreatedListener extends AbstractController
{
    /**
     * @var RequestStack
     */
    private $requestStack;

    /**
     * @param RequestStack $requestStack
     */
    public function __construct(RequestStack $requestStack)
    {
        $this->requestStack = $requestStack;
    }

    /**
     * @param JJWTCreatedEvent $event
     */
    public function onJWTCreated(JJWTCreatedEvent $event)
    {
        $request = $this->requestStack->getCurrentRequest();

        // Récupérer le User-Agent de la requête client
        $userAgent = $request->headers->get('User-Agent');

        // Récupérer les données et les en-têtes du JWT
        $payload = $event->getData();
        $header = $event->getHeader();

        // Ajouter le User-Agent au payload si vous en avez besoin
        $payload['ip'] = $request->getClientIp();
        // Ajouter le User-Agent au header du JWT
        $payload['cty'] = "JWTTranquillo";
        $payload['User-Agent'] = $userAgent;

        // Définir les nouvelles données et en-têtes pour le JWT
        $event->setData($payload);
    }
}
```

```
final class UsersJWTCreatedListener extends AbstractController
{
    public function onJWTCreated(JJWTCreatedEvent $event)
    {
        // Définir les nouvelles données et en-têtes pour le JWT
        $event->setData($payload);
        // $event->setHeader($header);

        // Définir la date d'expiration du JWT
        $expiration = new \DateTime('+30 day');
        $expiration->setTime(2, 0, 0);
        $payload['exp'] = $expiration->getTimestamp();
        $event->setData($payload);
    }

    /**
     * @param JWTD decodedEvent $event
     */
    public function onJWTD decodedEvent(JWTD decodedEvent $event)
    {
        $request = $this->requestStack->getCurrentRequest();
        $userAgent = $request->headers->get('User-Agent');
        $userCty = $request->headers->get('cty');

        $payload = $event->getPayload();

        if (!isset($payload['ip']) || $payload['ip'] !== $request->getClientIp())
            $event->markAsInvalid();
        if (!isset($payload['cty']) || $payload['cty'] !== $userCty)
            $event->markAsInvalid();
        if (!isset($payload['User-Agent']) || $payload['User-Agent'] !==
            $event->markAsInvalid();
    }
}
```


C.3.2.2.2. Cross-Site Scripting (XSS) :

Les attaques XSS injectent des scripts malveillants dans les pages web consultées par d'autres utilisateurs.

Prévention :

- Échapper correctement les données sorties pour HTML, JavaScript, CSS et les attributs URL.
- Utiliser des Content Security Policies (CSP) pour restreindre les sources de contenu pouvant être chargées sur votre site.
- Valider et assainir les entrées utilisateur.

C.3.2.2.3. Cross-Site Request Forgery (CSRF) :

Les attaques CSRF exploitent la confiance d'un site web envers l'utilisateur, incitant celui-ci à effectuer des actions non désirées.

Prévention :

- Utiliser des tokens CSRF pour vérifier l'authenticité des requêtes provenant de l'utilisateur.
- Vérifier les en-têtes de la requête HTTP (comme Origin et Referer).
- Implémenter des mécanismes d'authentification robuste, comme l'authentification à deux facteurs (2FA).

C.3.2.2.4. Brute Force

Les attaques par force brute tentent de deviner les mots de passe ou les clés de chiffrement en essayant toutes les combinaisons possibles.

Prévention :

- Limiter le nombre de tentatives de connexion pour prévenir les attaques par force brute.
- Utiliser des CAPTCHA pour différencier les utilisateurs humains des scripts automatisés.
- Exiger des mots de passe complexes et utiliser des méthodes d'authentification supplémentaires comme 2FA.

C.3.2.2.5. Man-in-the-Middle (MITM)

Les attaques MITM interceptent et modifient les communications entre deux parties sans leur connaissance.

Prévention :

- Utiliser le chiffrement SSL/TLS pour sécuriser les communications entre les clients et le serveur.
- Mettre en place des certificats de sécurité robustes et assurer leur renouvellement régulier.
- Utiliser des protocoles sécurisés comme HTTPS, SSH, et VPN.

C.3.2.2.6. Déni de Service (DoS) et Déni de Service Distribué (DDoS)

Les attaques DoS et DDoS visent à rendre un service indisponible en submergeant le système avec un grand nombre de requêtes.

Prévention :

- Mettre en place des pare-feu applicatifs (WAF) et des systèmes de détection des intrusions (IDS).
- Utiliser des services de protection DDoS fournis par des tiers spécialisés.
- Implémenter des stratégies de limitation de débit et de gestion de la bande passante.

C.3.2.2.7. Vol de Session

Les attaques de vol de session exploitent des cookies de session pour prendre le contrôle de la session d'un utilisateur légitime.

Prévention :

- Utiliser des cookies de session sécurisés (Secure et HttpOnly).
- Régénérer les identifiants de session après une connexion réussie.
- Mettre en place des expirations de session et forcer la reconnexion après une période d'inactivité.

C.3.2.3. Service

```

serveur-backend > src > Service > TaskService.php > TaskService
16 class TaskService extends AbstractController
17 // #####
18 // ----- UPDATE -----
19 // #####
20
21 /**
22 * @param TaskDto $taskDto
23 * @param int|string|null $userId
24 * @return array
25 */
26 public function update(TaskDto $taskDto, int | string | null $userId)
27 {
28     $id = $taskDto->getId();
29     $task = $this->taskRepository->findOneByTask('id', $id);
30
31     /* Cet extrait de code vérifie si l'utilisateur essayant de mettre à jour une tâche est... */
32     if ($userId !== $task->getUsersId()) {
33         $title = "Access is unauthorized";
34         $message = "This is not your task";
35         return ["task" => null, "title" => $title, "code" => Response::HTTP_FORBIDDEN, "message" => $message];
36     }
37
38     /**
39      * ...
40      * $newName = $taskDto->getName();
41      * $newDescription = $taskDto->getDescription();
42      * $newReminder = $taskDto->getReminder();
43      * $newStartAt = $taskDto->getStartAt();
44      * $newEndAt = $taskDto->getEndAt();
45
46      * if ($newName !== null) {
47          $task->setName($newName);
48      }
49      * if ($newDescription !== null) {
50          $task->setDescription($newDescription);
51      }
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204

```

C.3.2.4. DTO

```

serveur-backend > src > Dto > TaskDto.php > TaskDto
1 <?php
2
3 namespace App\Dto;
4
5 // use App\Validator\DateTimeImmutable;
6
7 use App\Entity\User;
8 use App\Validator\TpaLength;
9 use App\Validator\UserRegex;
10 use Symfony\Component\Serializer\Annotation\Groups;
11 use Symfony\Component\Validator\Constraints as Assert;
12
13 class TaskDto
14 {
15     #[Groups(['tasks: read', 'tasks: create'])]
16     #[UserRegex(regex: 'number', entity: "task", field: "id")]
17     private ?int $id = null;
18
19     #[Groups(['tasks: read', 'tasks: create'])]
20     #[AssertNotBlank(message: 'error.task.name: The key « name » must be a non-empty string.')]
21     #[TpaLength(min: 5, max: 50, entity: "task", field: "name")]
22     private ?string $name = null;
23
24     #[Groups(['tasks: read', 'tasks: create'])]
25     #[AssertNoSuspiciousCharacters()]
26     private ?string $description = null;
27
28     #[Groups(['tasks: read', 'tasks: create'])]
29     #[UserRegex(regex: 'number', information: "Reminder task is calculated in minute(s)", entity: "task", field: "reminder")]
30     #[AssertPositiveOrZero()]
31     private ?int $reminder = null;
32
33     #[Groups(['tasks: read', 'tasks: create'])]
34     private ?\DateTimeImmutable $startAt = null;
35
36     #[Groups(['tasks: read', 'tasks: create'])]
37     private ?\DateTimeImmutable $endAt = null;
38
39     #[Groups(['tasks: read', 'tasks: create'])]
40     private ?\DateTimeImmutable $createdAt = null.
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93

```


C.4. Base de Données

C.4.1. Code SQL

```

AppFixtures.php          02_DLL_schema.sql M
database > sql > 02_DLL_schema.sql > SET NAMES utf8
SET NAMES utf8;
SET time_zone = `+01:00`;
SET foreign_key_checks = 0;
SET NAMES utf8mb4;
CREATE DATABASE IF NOT EXISTS tranquillo
/*I40100 DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci */;

USE tranquillo;
DROP TABLE IF EXISTS tpa_subtasks,
tpa_tasks,
tpa_users,
tpa_roles;
CREATE TABLE
IF NOT EXISTS tpa_users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(180) NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    firstname VARCHAR(50) NOT NULL,
    user_password VARCHAR(255) NOT NULL,
    user_role VARCHAR(255) NULL,
    user_create_at DATETIME NOT NULL DEFAULT
    CURRENT_TIMESTAMP,
    CONSTRAINT tpa_users_ukey UNIQUE (email),
    INDEX tpa_users_email_ikey (email),
    INDEX tpa_users_lastname_ikey (lastname),
    INDEX tpa_users_firstname_ikey (firstname),
    INDEX tpa_users_create_at_ikey (user_create_at)
) ENGINE = InnoDB;

CREATE TABLE
IF NOT EXISTS tpa_tasks (
    task_id INT AUTO_INCREMENT PRIMARY KEY,
    task_name VARCHAR(50) NOT NULL,
    task_description TEXT,
    task_reminder INT DEFAULT NULL,
    task_start_at DATETIME NULL,
    task_end_at DATETIME NULL,
    task_create_at DATETIME NOT NULL DEFAULT
    CURRENT_TIMESTAMP,
    users_id INT NOT NULL,
    CONSTRAINT tpa_tasks_users_fkey FOREIGN KEY
    (users_id) REFERENCES tpa_users (user_id),
    CONSTRAINT tpa_tasks_ukey UNIQUE (task_name,
    task_create_at),
    INDEX tpa_tasks_name_ikey (task_name),
    INDEX tpa_tasks_create_at_ikey (task_create_at)
) ENGINE = InnoDB;

```

C.4.2. Compte utilisateur MariaDB pour l'API

Éditer les privilèges : Compte d'utilisateur 'api'@'%'

Base de données	Priviléges	Grant	Priviléges spécifiques à une table	Action
tranquillo	SELECT, INSERT, UPDATE, DELETE	Non	Non	Éditer les priviléges Révoquer

Ajouter des priviléges sur ces bases de données :

Exécuter

```

serveur-backend > .env.local
43  #
44  # Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html#connecting-using-a-url
45  # IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
46  #
47  DATABASE_URL="mysql://${MYSQL_USER_API}:${MYSQL_PASSWORD_API}@${MYSQL_LOCALHOST}: ${SQL_LOCALHOST_PORT}/${
    MYSQL_DATABASE}?serverVersion=${MARIADB_VERSION}-MariaDB&charset=utf8mb4"
48  ###< doctrine/doctrine-bundle ####
49

```

C.4.3. Fixtures

C.4.3.1. Base

EXPLORATEUR

```

TRANQUILLO
> .github
> .vscode
> BACKUP
< database
> < sql
> < sql_test
  03_DML_referencial.sql
  04_DML_test.sql
  fixturesSubtasks.php
  fixturesTasks.php
  fixturesUsers.php
  tranquillo_mariadb
> docs
> install
> serveur-backend
  > bin
  > config
    > config
  > jwt
  > packages
    ! api_platform.yaml
    ! cache.yaml
    ! doctrine_migrations.yaml
    ! doctrine.yaml
    ! framework.yaml
    ! lexik_jwt_authentication...
    ! nelmio_cors.yaml
    ! routing.yaml
    ! security.yaml
    ! translation.yaml
    ! twig.yaml
    ! validator.yaml
  > routes
  fixtures.php
  preload.php
  ! routes.yaml

```

fixturesUsers.php

```

1 <?php
2 $fixturesUsers = array(
3   array(
4     'role' => 'ROLE_WEBMASTER',
5     'email' => 'contact@corbisier.fr',
6     'lastname' => 'CORBISIER',
7     'firstname' => 'Eric',
8     'password' => 'Eric:CORBISIER:1234'
9   ),
10  array(
11    'role' => 'ROLE_ADMIN',
12    'email' => 'ecorbisier.simplon@gmail.com',
13    'lastname' => 'CORBISIER',
14    'firstname' => 'Eric',
15    'password' => 'Eric:CORBISIER:1234'
16  ),
17  array(
18    'role' => 'ROLE_USER',
19    'email' => 'jean.dupont@example.com',
20    'lastname' => 'Dupont',
21    'firstname' => 'Jean',
22    'password' => 'Jean:Dupont:1234'
23  ),
24  array(
25    'role' => 'ROLE_USER',
26    'email' => 'marie.durand@example.com',
27    'lastname' => 'Durand',
28    'firstname' => 'Marie',
29    'password' => 'Marie:Durand:1234'
30  ),
31  array(
32    'role' => 'ROLE_USER',
33    'email' => 'pierre.martin@example.com',
34    'lastname' => 'Martin',
35    'firstname' => 'Pierre',
36    'password' => 'Pierre:Martin:1234'
37  ),
38  array(
39    'role' => 'ROLE_USER',
40    'email' => 'sophie.lefebvre@example.com',
41    'lastname' => 'Lefebvre'
42 )

```

fixturesTasks.php

```

1 <?php
2 $fixturesTasks = array(
3   array(
4     'usersEmail' => 'jean.dupont@example.com',
5     'taskTitle' => 'Réunion de projet',
6     'taskDescription' => 'Réunion pour discuter de l'avancement du projet',
7     'taskReminder' => NULL,
8     'taskStartAt' => '2024-04-17 09:00:00',
9     'taskEndAt' => '2024-04-17 10:30:00'
10  ),
11  array(
12    'usersEmail' => 'marie.durand@example.com',
13    'taskTitle' => 'Présentation client',
14    'taskDescription' => 'Préparer la présentation pour le client',
15    'taskReminder' => NULL,
16    'taskStartAt' => '2024-04-18 14:00:00',
17    'taskEndAt' => '2024-04-18 16:00:00'
18  ),
19  array(
20    'usersEmail' => 'pierre.martin@example.com',
21    'taskTitle' => 'Analyse de données',
22    'taskDescription' => 'Analyser les données de performance',
23    'taskReminder' => NULL,
24    'taskStartAt' => '2024-04-19 10:00:00',
25    'taskEndAt' => '2024-04-19 12:00:00'
26  ),
27  array(
28    'usersEmail' => 'sophie.lefebvre@example.com',
29    'taskTitle' => 'Révision des documents',
30    'taskDescription' => 'Revoir et corriger les documents',
31    'taskReminder' => NULL,
32    'taskStartAt' => '2024-04-20 09:30:00',
33    'taskEndAt' => '2024-04-20 11:30:00'
34  ),
35  array(
36    'usersEmail' => 'michel.dubois@example.com',
37    'taskTitle' => 'Formation sur les nouvelles technologies',
38    'taskDescription' => 'Participer à la formation sur les nouvelles technologies',
39    'taskReminder' => NULL,
40    'taskStartAt' => '2024-04-21 13:00:00'
41 )

```

C.4.3.2. Utilisation

EXPLORATEUR

```

TRANQUILLO
< serveur-backend
  < config
    ! routes.yaml
    ! services.yaml
  > migrations
  > public
    > bundles
    fixtures.php
  > src
    > ApiResource
    > Controller
      & .gitignore
        HomeController.php
        TaskAdminController.php
        TaskController.php M
        UserAdminController.php
        UserController.php
      > DataFixtures
        AppFixtures copy.php
        AppFixtures.php
      < Dto
        TaskDto.php M
        UserDto.php
      < Entity
        & .gitignore
          Task.php
          User.php
      < EventListener
        UserExceptionListener.php
        UsersJWTCreatedListener...
        UsersJWTResponse...
      > Helper
      > Repository
      > Security
      < Service
        CLOUDFRONT

```

fixturesUsers.php

```

12 /**
13  * @package App\DataFixtures */
14 class AppFixtures extends Fixture
15 {
16   private $userPasswordHasher;
17
18   public function __construct(UserPasswordEncoderInterface $userPasswordHasher)
19   {
20     $this->userPasswordHasher = $userPasswordHasher;
21   }
22
23   /* La méthode "public function load(ObjectManager): void" dans la classe "AppFixtures" est...
24   public function load(ObjectManager $manager): void
25   {
26     // Récupérer le chemin absolu du dossier courant...
27     $cheminCourant = __DIR__;
28     // Diviser le chemin en segments en utilisant le délimiteur "/"
29     $segments = explode("/", $cheminCourant);
30     $folderBack = $ENV['FOLDER_BACK'];
31     $folderData = $ENV['FOLDER_DATA'];
32     // Trouver l'index du segment contenant "triquillo"
33     $indexFolderBack = array_search($folderBack, $segments);
34     // Reconstituer le chemin absolu du dossier triquillo en utilisant les segments précédents
35     $cheminBack = implode("/", array_slice($segments, 0, $indexFolderBack + 1));
36     $cheminData = dirname($cheminBack) . "/" .
37     $folderData . "/sql_test";
38
39     // #####
40     require_once($cheminData . "/fixturesUsers.php");
41
42     foreach ($fixturesUsers as $fixUser) {
43       $user = new User();
44       // Accéder aux éléments de l'utilisateur
45       $user->setRoles([$fixUser['role']]);
46       $user->setEmail($fixUser['email']);
47     }
48   }
49
50   foreach ($fixturesUsers as $fixUser) {
51     $user = new User();
52     // Accéder aux éléments de l'utilisateur
53     $user->setRoles([$fixUser['role']]);
54     $user->setEmail($fixUser['email']);
55   }
56 }

```

AppFixtures.php

```

14 class AppFixtures extends Fixture
15 {
16   public function load(ObjectManager $manager): void
17   {
18     foreach ($fixturesTasks as $fixTask) {
19       $task = new Task();
20       // Accéder aux éléments de la tâche
21       $userEmail = $fixTask['usersEmail'];
22
23       // Recherche de l'utilisateur par son email dans la liste des utilisateurs
24       $filteredUsers = array_filter($listUsers,
25         function ($user) use ($userEmail) {
26           return $user->getEmail() === $userEmail;
27         });
28
29       // Accéder aux éléments de la tâche
30       $users = reset($filteredUsers); // Récupérer le premier utilisateur correspondant
31       $userId = $users->getId(); // Récupérer l'ID de l'utilisateur
32
33       // Utiliser l'entité utilisateur dans la tâche
34       $task->setUserId($userId);
35
36       $task->setName($fixTask['taskTitle']);
37       $task->setDescription($fixTask['taskDescription']);
38       $task->setReminder($fixTask['taskReminder']);
39
40       $task->setCreateAt(new \DateTimeImmutable());
41
42       $taskStartAtString = $fixTask['taskStartAt'];
43       $taskStartAt = new \DateTimeImmutable(
44         $taskStartAtString);
45       $task->setStartAt($taskStartAt);
46
47       $taskEndAtString = $fixTask['taskEndAt'];
48       $taskEndAt = new \DateTimeImmutable(
49         $taskEndAtString);
50       $task->setEndAt($taskEndAt);
51
52       $manager->persistent($task);
53       $manager->flush();
54     }
55   }
56 }

```



C.4.4. Users

C.4.4.1. Architecture

Schéma de la base de données :

```

Nouvelle base de données
  |
  +-- information_schema
  +-- mysql
  +-- performance_schema
  +-- sys
  +-- tranquillo
    +-- Nouvelle table
      +-- tpa_subtasks
      +-- tpa_tasks
        +-- Colones
        +-- Index
      +-- tpa_users
        +-- Colones
        +-- Nouvelle colonne
          +-- email (UNI, varchar)
        +-- lastname (MUL, varchar)
        +-- firstname (MUL, varchar)
        +-- user_create_at (MUL, varchar)
        +-- user_id (PRI, int)
        +-- user_password (varchar)
        +-- user_role (varchar, NU)
        +-- Index

```

Structure de table pour la table `tpa_users` :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	<code>user_id</code>	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
2	<code>email</code>	varchar(180)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
3	<code>lastname</code>	varchar(50)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
4	<code>firstname</code>	varchar(50)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
5	<code>user_password</code>	varchar(255)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
6	<code>user_role</code>	varchar(255)	utf8mb4_general_ci		Oui	NULL			Modifier Supprimer Plus
7	<code>user_create_at</code>	datetime			Non	current_timestamp()			Modifier Supprimer Plus

Index pour la table `tpa_users` :

Action	Nom de l'index	Type	Unique	Comprimé	Colonne	Cardinalité	Interclassement	Null	Commentaire
Éditer Supprimer	PRIMARY	BTREE	Oui	Non	<code>user_id</code>	19	A	Non	
Éditer Supprimer	<code>tpa_users_ukey</code>	BTREE	Oui	Non	<code>email</code>	19	A	Non	
Éditer Supprimer	<code>tpa_users_email_ikey</code>	BTREE	Non	Non	<code>email</code>	19	A	Non	
Éditer Supprimer	<code>tpa_users_lastname_ikey</code>	BTREE	Non	Non	<code>lastname</code>	19	A	Non	
Éditer Supprimer	<code>tpa_users_firstname_ikey</code>	BTREE	Non	Non	<code>firstname</code>	19	A	Non	
Éditer Supprimer	<code>tpa_users_create_at_ikey</code>	BTREE	Non	Non	<code>user_create_at</code>	19	A	Non	

C.4.4.2. Valeur

Tableau de données pour la table `tpa_users` :

	<code>user_id</code>	<code>email</code>	<code>lastname</code>	<code>firstname</code>	<code>user_password</code>	<code>user_role</code>	<code>user_create_at</code>
1	25	contact@corbisier.fr	CORBISIER	Eric	\$2y\$13\$Z1aOGECmKSAJuZPtLyNVeD8OrGsmV5gD6Ryz3UGK0... ["ROLE_WEBMASTER"]	["ROLE_ADMIN"]	2024-05-06 12:50:50
2	26	eric.corbisier.simplon@gmail.com	CORBISIER	Eric	\$2y\$13\$QB3tNoLNUrJ2HCVIK/VeOErf7AcG3WsYR5QLn/23UP...	["ROLE_ADMIN"]	2024-05-06 12:50:51
3	27	jean.dupont@example.com	Dupont	Jean	\$2y\$13\$OYWhpOxelZPQaPdzGkc4eyDEWuRtGopSjQjlthVdw...	["ROLE_USER"]	2024-05-06 12:50:52
4	28	marie.durand@example.com	Durand	Marie	\$2y\$13\$B8sYiuZOnQhqAWWR/Yu.p23gNnfFcwbePflUvBj9D...	["ROLE_USER"]	2024-05-06 12:50:52
5	29	pierre.martin@example.com	Martin	Pierre	\$2y\$13\$NYXPXlPIManhoxc3fGbjAiegMTISVnrtQzs5vY4...	["ROLE_USER"]	2024-05-06 12:50:53
6	30	sophie.lefebvre@example.com	Lefebvre	Sophie	\$2y\$13\$rh9bCOloDGm47ONg9WGvitolQkoZcrZlFzbqv2FZWL...	["ROLE_USER"]	2024-05-06 12:50:53
7	31	michel.dubois@example.com	Dubois	Michel	\$2y\$13\$kuF_ZFl5NlusRtU3Pj.kw7zSEFy085xnYfLxe...	["ROLE_USER"]	2024-05-06 12:50:54
8	32	isabelle.roux@example.com	Roux	Isabelle	\$2y\$13\$HrG92jPrpwFYHvpSCPTO/O2Kxe1IGktO4RhzwTly...	["ROLE_USER"]	2024-05-06 12:50:55
9	33	philippe.leclerc@example.com	Leclerc	Philippe	\$2y\$13\$wges91SnDeyaJemCoCpuYQans/ObNjdDNZmwCFLWK...	["ROLE_USER"]	2024-05-06 12:50:55
10	34	sylvie.garcia@example.com	Garcia	Sylvie	\$2y\$13\$yWL3mhBznTrzAx88wqnuzugIMXHOdNqaRZRBXF0d...	["ROLE_USER"]	2024-05-06 12:50:56
11	35	david.fournier@example.com	Fournier	David	\$2y\$13\$D4TSeLaQ8cu.Jd.7sDyOXOTQLbpLcrWtaI0J6AU6p...	["ROLE_USER"]	2024-05-06 12:50:56
12	36	anne.legrand@example.com	Legrand	Anne	\$2y\$13\$U59epDyCtnfCavn1Lap4V1FdubTCUQ0Qfipn9pv...	["ROLE_USER"]	2024-05-06 12:50:57

C.4.5. Tasks

C.4.5.1. Architecture

The screenshot shows the phpMyAdmin interface for the 'tpa_tasks' table. The left sidebar shows the database structure with 'tpa_tasks' selected. The main area displays the table structure with columns: task_id, task_name, task_description, task_reminder, task_start_at, task_end_at, task_create_at, and users_id. Below the table structure is an 'Index' section showing four indexes: PRIMARY, tpa_tasks_ukey, tpa_tasks_users_fkey, and tpa_tasks_name_ikey. At the bottom, there is a button to create a new index.

C.4.5.2. Valeur

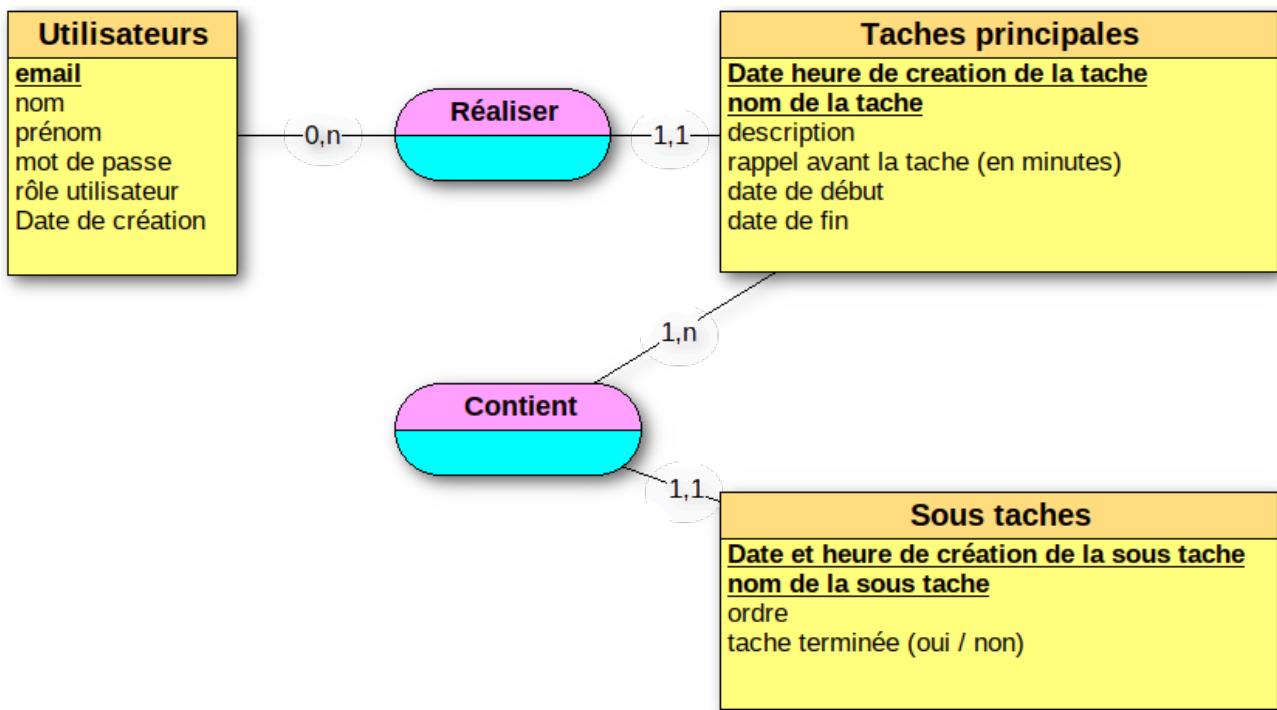
The screenshot shows the data in the 'tpa_tasks' table. The table has 12 columns: task_id, task_name, task_description, task_reminder, task_start_at, task_end_at, task_create_at, users_id, and several timestamp columns. The data includes tasks like 'Réunion de projet', 'Présentation client', and 'Formation sur les nouvelles technologies'. The last row shows a task with id 95.

C.4.5.3. Relationnel

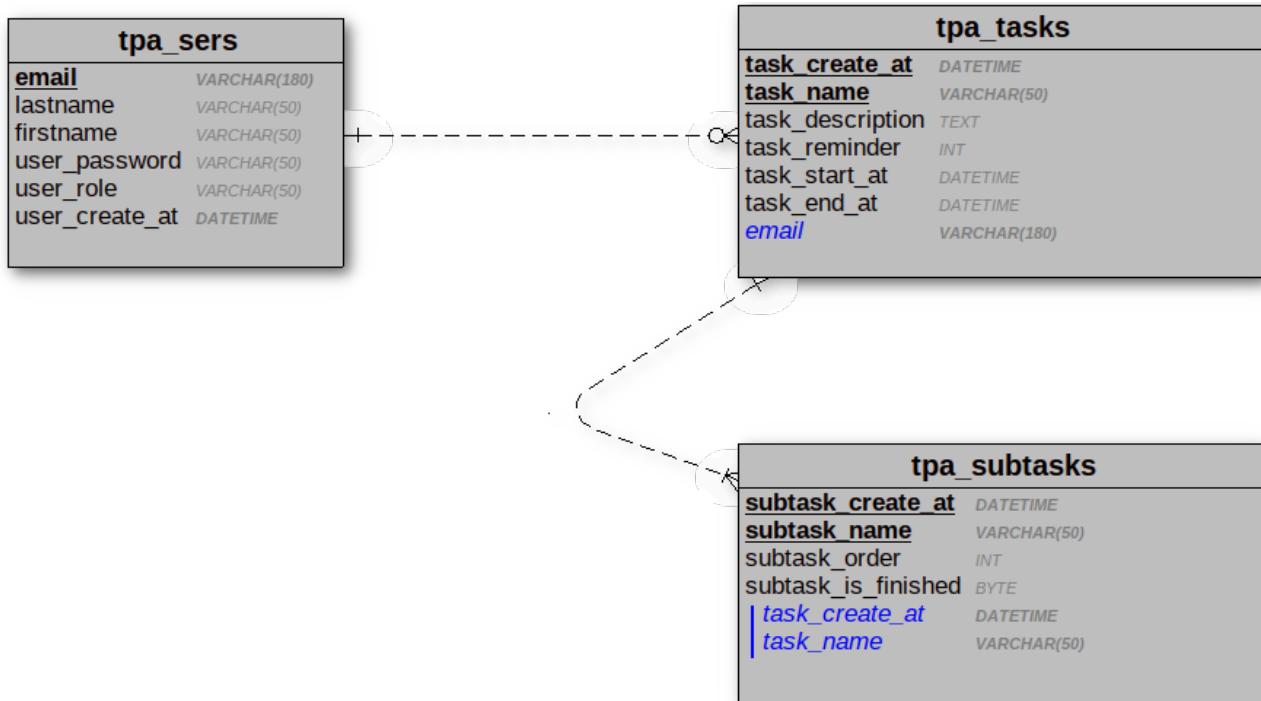
The screenshot shows the foreign key constraints for the 'tpa_tasks' table. It defines a constraint named 'tpa_tasks_users_fkey' that links the 'users_id' column to the 'user_id' column in the 'tpa_users' table. The constraint uses the 'RESTRICT' rule for both ON DELETE and ON UPDATE.

D.Les Diagrammes

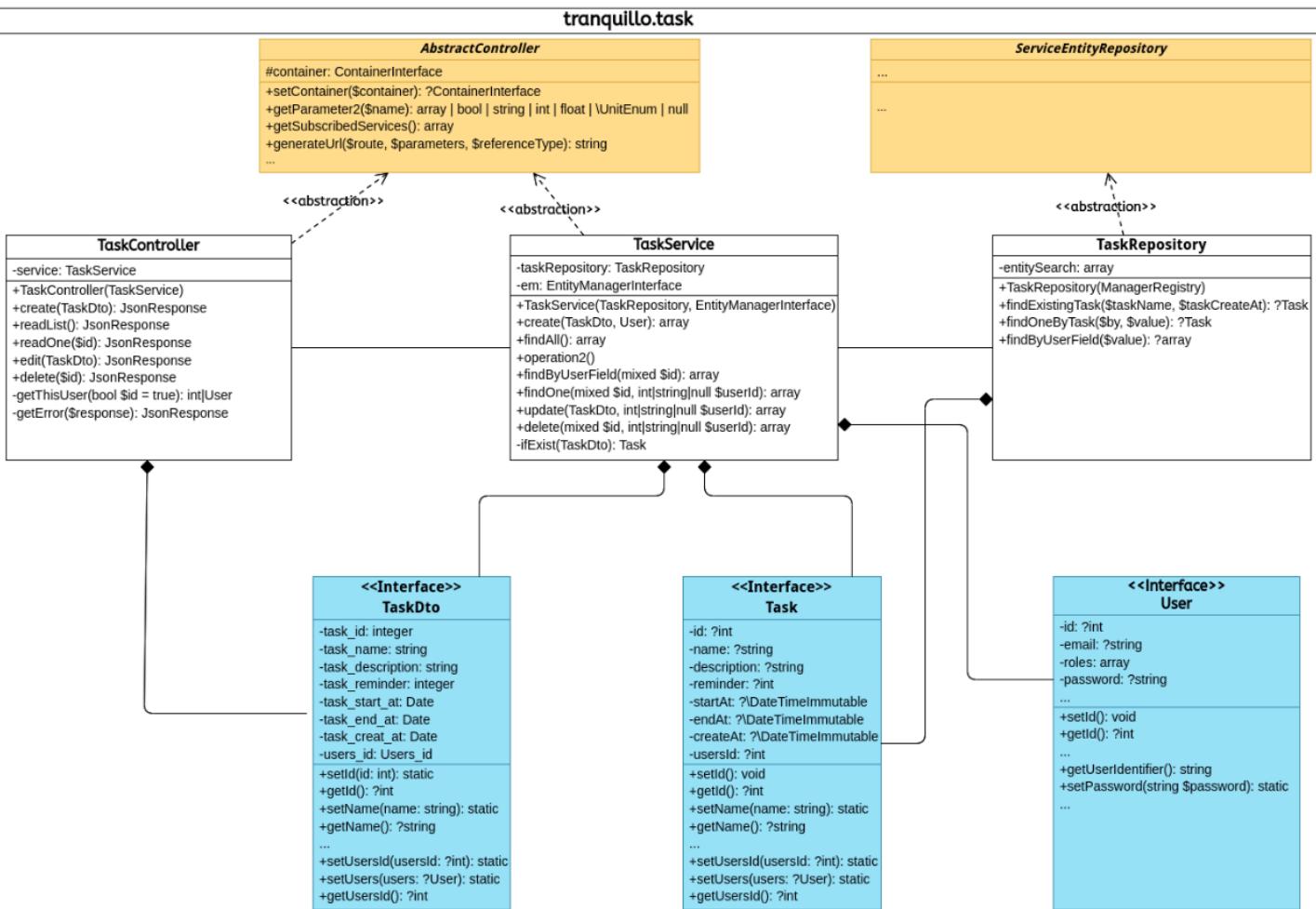
D.1. MCD



D.2. MLD



D.3. Classe



Task.php M TaskController.php M

```

server-backend > src > Controller > TaskController.php > TaskController
1 <?php
2
3 namespace App\Controller;
4
5 use App\Dto\TaskDto;
6 use App\Entity\User;
7 use App\Helper\ObjectHydrator;
8 use App\Service\TaskService;
9 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController
10 use Symfony\Component\HttpFoundation\Request
11 use Symfony\Component\HttpFoundation\Response
12 use Symfony\Component\HttpKernel\Attribute\ServiceSubscriberInterface
13 use Symfony\Component\Routing\Annotation\Route
14 use Symfony\Component\Routing\Request
15 use Symfony\Component\Security\Http\Symfony\Bundle\FrameworkBundle\Controller\AbstractController
16 use Symfony\Component\HttpKernel\Attribute\AsController
17
18 #[AsController]
19 #[Route('api/task', name: 'app_task')]
20 final class TaskController extends AbstractController
21 {
22     private $service;
23
24     /**
25      * ----- create -----
26      * @return JsonResponse
27      */
28
29     #[Route(['', '/'], name: 'task_create', methods: ['POST'])]
30     public function create#[MapRequestPayload](serializationContext: ['tasks': c
31     {
32         $response = $this->service->create($taskDto, $this->getThisUser($id: fal
33
34         if ($response['task'] == null) {
35             return $this->getError($response);
36
37         $response = $this->service->findOne($response['task'], $this->getThisUs
38
39         $codeHttp = intval($response['code']);
40
41         if ($codeHttp != 201) {
42             return $this->error($codeHttp, $response['message']);
43
44         $codeHttp = 201;
45
46         if ($codeHttp == 201) {
47             return $this->success($response);
48
49         }
50
51         $codeHttp = 200;
52
53         if ($codeHttp == 200) {
54             return $this->success($response);
55
56         }
57     }
58
59     /**
60      * ----- read -----
61      * @param string $id
62      * @return JsonResponse
63      */
64
65     #[Route('/{id}', name: 'task_read')]
66     public function read(string $id): JsonResponse
67     {
68         $task = $this->service->findOne($id);
69
70         if ($task == null) {
71             return $this->error(404, 'Task not found');
72
73         }
74
75         return $this->success($task);
76     }
77
78
79     /**
80      * ----- update -----
81      * @param string $id
82      * @param array $data
83      * @return JsonResponse
84      */
85
86     #[Route('/{id}', name: 'task_update')]
87     public function update(string $id, array $data): JsonResponse
88     {
89         $task = $this->service->findOne($id);
90
91         if ($task == null) {
92             return $this->error(404, 'Task not found');
93
94         }
95
96         $task->update($data);
97
98         return $this->success($task);
99     }
100
101
102     /**
103      * ----- delete -----
104      * @param string $id
105      * @return JsonResponse
106      */
107
108     #[Route('/{id}', name: 'task_delete')]
109     public function delete(string $id): JsonResponse
110     {
111         $task = $this->service->findOne($id);
112
113         if ($task == null) {
114             return $this->error(404, 'Task not found');
115
116         }
117
118         $task->delete();
119
120         return $this->success();
121     }
122 }
  
```

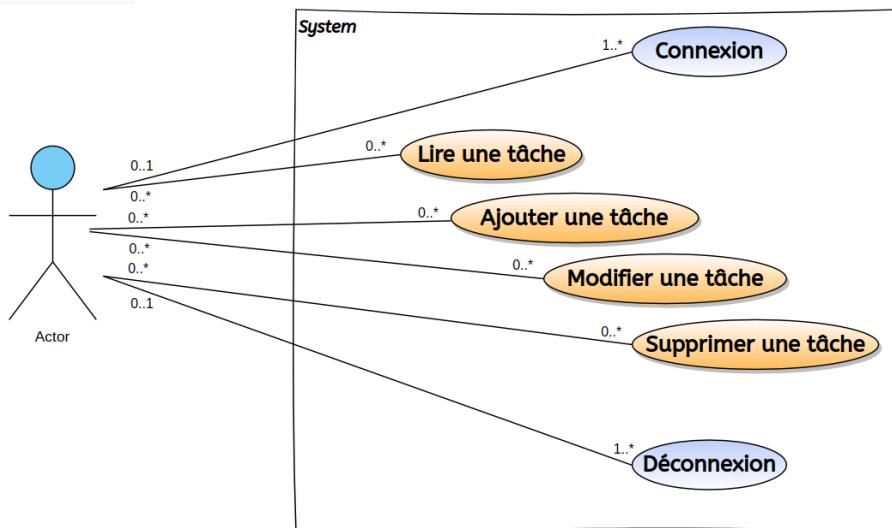
AbstractController.php M

```

server-backend > vendor > symfony > framework-bundle > Controller > AbstractController.php > AbstractController
42 use Symfony\Component\Security\Csrftoken\Manager\TokenManagerInterface;
43 use Symfony\Component\Serializer\SerializerInterface;
44 use Symfony\Component\WebLink\EventListener\AddLinkHeaderListener;
45 use Symfony\Component\WebLink\GenericLinkProvider;
46 use Symfony\Component\WebLink\Header\Header;
47 use Symfony\Contracts\Service\Attribute\Required;
48 use Symfony\Contracts\Service\SubscriberInterface;
49 use Twig\Environment;
50
51 /**
52 * Provides shortcuts for HTTP-related features in controllers.
53 */
54 #[@author Fabien Potencier <fabien@symfony.com>]
55 */
56 abstract class AbstractController implements ServiceSubscriberInterface
57 {
58     protected ContainerInterface $container;
59
60     #[Required]
61     public function setContainer(ContainerInterface $container): ?
62     ContainerInterface
63     {
64         $previous = $this->container ?? null;
65         $this->container = $container;
66
67         return $previous;
68     }
69
70     /**
71      * Gets a container parameter by its name.
72      */
73     protected function getParameter(string $name): array|bool|string|int
74     float|\UnitEnum|null
75     {
76         if (!($this->container->has('parameter_bag'))) {
77             throw new ServiceNotFoundException('parameter_bag', null, null,
78             []);
79             $this->container->get('parameter_bag')->get($name);
80         }
81     }
82
83     /**
84      * Gets a container parameter by its name.
85      */
86     protected function getOptionalParameter(string $name): mixed
87     {
88         if (!($this->container->has('parameter_bag'))) {
89             throw new ServiceNotFoundException('parameter_bag', null, null,
90             []);
91             $this->container->get('parameter_bag')->get($name);
92         }
93     }
94
95     /**
96      * Gets a container parameter by its name.
97      */
98     protected function hasParameter(string $name): bool
99     {
100         if (!($this->container->has('parameter_bag'))) {
101             throw new ServiceNotFoundException('parameter_bag', null, null,
102             []);
103             $this->container->get('parameter_bag')->has($name);
104         }
105     }
106
107     /**
108      * Registers a service subscriber.
109      */
110     public function addServiceSubscriber(ServiceSubscriberInterface $subscriber): void
111     {
112         $this->container->addEventSubscriber($subscriber);
113     }
114
115     /**
116      * Adds an event listener.
117      */
118     public function addEventListener(EventListenerInterface $listener): void
119     {
120         $this->container->addEventSubscriber($listener);
121     }
122 }
  
```



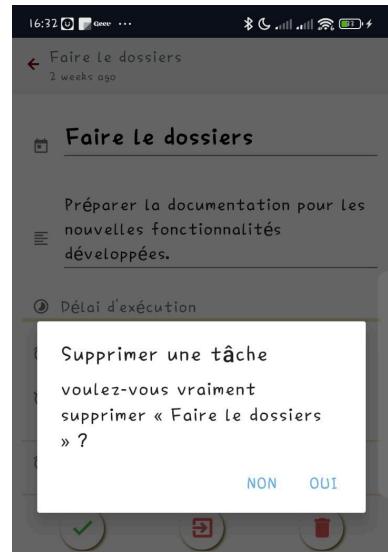
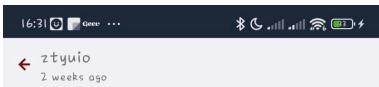
D.4. Use Case



E. Les Tests

E.1. Interface Utilisateurs





E.2. Couche métier

Les données sont valide et la réponse attend également

Status: 200 OK Size: 1.28 KB Time: 1.51 s

Test	Result
Response Code equal to 200	Pass
Response Body is JSON	Pass
Response Body contains "token"	Pass
Header Content-Type equal to application/json	Pass
Response Body contains "code": 202	Pass
Response Body contains "user":	Pass

Les données sont invalide mais la réponse attendue est valide

Status: 403 Forbidden Size: 135 Bytes Time: 44 ms

Test	Result
Response Code equal to 200 => Actual: 403	Fail
Response Body is JSON	Pass
Response Body contains "status": "Access forbidden", "message": "Incorrect credentials, please check that your username/password is set correctly.", "code": 403	Fail

Les données sont invalide et la réponse attendue également

Status: 403 Forbidden Size: 135 Bytes Time: 39 ms

Test	Result
Response Code equal to 200 => Actual: 403	Fail
Response Body is JSON	Pass
Response Body contains "status": "Access forbidden", "message": "Incorrect credentials, please check that your username/password is set correctly.", "code": 403	Fail
Header Content-Type equal to application/json	Pass
Response Body contains "code": 202 => Actual: { "status": "Access forbidden", "message": "Incorrect credentials, please check that your username/password is set correctly.", "code": 403 }	Fail
Response Body contains "user": => Actual: { "status": "Access forbidden", "message": "Incorrect credentials, please check that your username/password is set correctly.", "code": 403 }	Fail

E.3. Tests Unitaire

E.3.1. Le Code

```

EXPLORATEUR
TRANQUILLO
  serveur-backend
    src
      Dto
        TaskDto.php
        UserDto.php
        Entity
        EventListener
        Helper
        Repository
          .gitignore
          TaskRepository.php
          UserRepository.php
        Security
        Service
          TaskService.php
          UserService.php
        Validator
          TpLength.php
          TpLengthValidator.php
          UserRegex.php
          UserRegexValidator.php
          Kernel.php
        templates
        tests
          Dto
            TaskDtoTest.php
        Entity
        RegisterTest.php
        bootstrap.php
        translations
        var
        vendor
          api-platform

```

```

TaskDtoTest.php U X RegisterTest.php U TaskDto.php M UserRegexValidator.php UserRegex.php A
serveur-backend > tests > Dto > TaskDtoTest.php > TaskDtoTest > testSetNameWithValidCharacters
1 <?php
2
3 namespace App\Tests\Dto;
4
5 use App\Dto\TaskDto;
6 use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;
7 use Symfony\Component\Validator\Validator\ValidatorInterface;
8
9 class TaskDtoTest extends KernelTestCase
{
10     private ValidatorInterface $validator;
11
12     protected function setUp(): void
13     {
14         self::bootKernel();
15         $this->validator = static::getContainer()->get(ValidatorInterface::class);
16     }
17
18     public function testSetNameWithValidCharacters()
19     {
20         $taskDto = new TaskDto();
21         $taskDto->setName('Test Task');
22
23         $errors = $this->validator->validate($taskDto);
24         $this->assertCount(0, $errors, (string) $errors);
25
26     }
27
28     public function testSetNameWithInvalidCharacters()
29     {
30         $taskDto = new TaskDto();
31         $taskDto->setName('Test Task!@#');
32
33         $errors = $this->validator->validate($taskDto);
34
35         $this->assertGreaterThanOrEqual(1, $errors);
36
37     }
38 }

```

E.3.2. Résultats

❖ Eronné

```

> eric@eric-ThinkPad-T450 ~/Documents/SIMPLON/tranquillo/serveur-backend (main) $ ./bin/phpunit --filter testSetNameWithValidCharacters
PHPUnit 11.1.3 by Sebastian Bergmann and contributors.

Runtime:      PHP 8.3.7
Configuration: /home/eric/Documents/SIMPLON/tranquillo/serveur-backend/phpunit.xml.dist

F
1 / 1 (100%)

Time: 00:00.152, Memory: 22.00 MB

There was 1 failure:

1) App\Tests\Dto\TaskDtoTest::testSetNameWithValidCharacters
Object(App\Dto\TaskDto).name:
    error.task.name: La valeur ne répond pas aux critères de sécurité ! (code de1e3db3-5ed4-4941-aae4-59f3667cc3a3)

Failed asserting that actual size 1 matches expected size 0.

/home/eric/Documents/SIMPLON/tranquillo/serveur-backend/tests/Dto/TaskDtoTest.php:26
/home/eric/Documents/SIMPLON/tranquillo/serveur-backend/vendor/phpunit/phpunit:104
-- 
There was 1 risky test:

1) App\Tests\Dto\TaskDtoTest::testSetNameWithValidCharacters
Test code or tested code did not remove its own exception handlers
/home/eric/Documents/SIMPLON/tranquillo/serveur-backend/tests/Dto/TaskDtoTest.php:19

FAILURES!
Tests: 1, Assertions: 1, Failures: 1, Warnings: 1, Risky: 1.

```

★ Valide après correction du regex de contrôle

```

> eric@eric-ThinkPad-T450 ~/Documents/SIMPLON/tranquillo/serveur-backend (main) $ ./bin/phpunit --filter testSetNameWithValidCharacters
PHPUnit 11.1.3 by Sebastian Bergmann and contributors.

Runtime:      PHP 8.3.7
Configuration: /home/eric/Documents/SIMPLON/tranquillo/serveur-backend/phpunit.xml.dist

R
1 / 1 (100%)

Time: 00:00.135, Memory: 22.00 MB

There was 1 risky test:

1) App\Tests\Dto\TaskDtoTest::testSetNameWithValidCharacters
Test code or tested code did not remove its own exception handlers
/home/eric/Documents/SIMPLON/tranquillo/serveur-backend/tests/Dto/TaskDtoTest.php:19

OK, but there were issues!
Tests: 1, Assertions: 1, Risky: 1.

```

➤ Le Regex

```

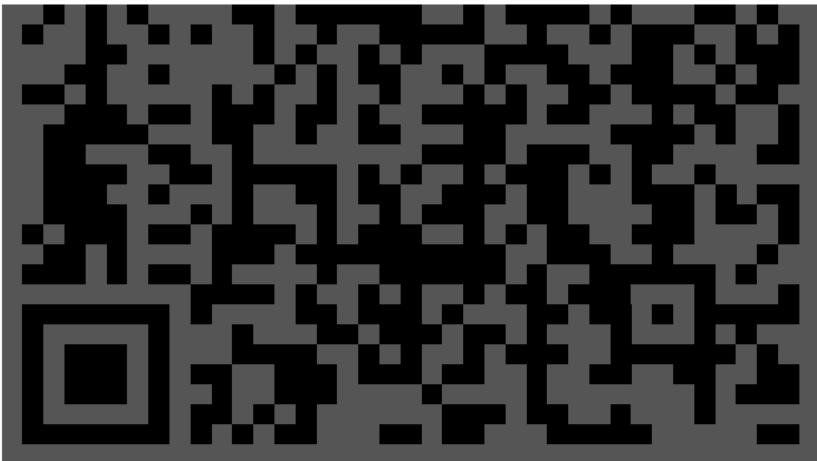
35   array $options = []
36   ) {
37     $pattern = ($regex === "password") ? '/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@#$%^&;$+=!=])(!.*[\ç<"">\mu`~\\\]).{0,}$/' : null;
38     $pattern = ($regex === "name") ? '/^[_a-zA-ZÀ-ӮӰӶӸ]+(_[-_a-zA-ZÀ-ӮӰӶӸ]+\s+)*[_a-zA-ZÀ-ӮӰӶӸ]\d*$/' : $pattern;
39     $pattern = ($regex === "number") ? '/^[\d-]+$/' : $pattern;
40

```

F.Le Déploiements

F.1. Déploiement Interface Utilisateur

PROBLÈMES 1 SORTIE TERMINAL PORTS POSTMAN CONSOLE COMMENTAIRES CONSOLE DE DÉBOGAGE



i Scan the above QR code with your device to connect to this session.
Scanning can be done with the default camera app on iOS or any other application capable of scanning QR codes.
i This CLI is interactive, you can press the following keys any time (make sure the terminal has focus):
i [R] - restart the Preview app on all connected devices
i [S] - reset the Preview app on all connected devices
i [O] - open the Web UI to scan the QR code
i Waiting for device connections...
Device connected: 22101316G [android: 2.0.7 (31)]
• Building for android [progress bar] building (54%) 4/5 entries 1399/1399 dependencies 425/426 modules 1 active
• Building for android [progress bar] sealing (89%)
/home/eric/Documents/SIMPLON/tranquillo/tranquilloApp/node_modules/@nativescript/preview-cli/dist/NativeScriptPreviewWebpackPlugin.js:2677
throw createError(FENOENT "stat" filename);

F.2. Déploiement Serveur Métier

PROBLÈMES 1 SORTIE TERMINAL PORTS POSTMAN CONSOLE COMMENTAIRES CONSOLE DE DÉBOGAGE

Lancement du back : ip_server= localhost
Docker desktop est lancé

localhost

Lien pour ouvrir symfony (CTRL + clic):
<http://localhost:8088>
[Tue May 28 21:04:44 2024] PHP 8.3.7 Development Server (http://localhost:8088) started
[Tue May 28 21:05:04 2024] 127.0.0.1:47086 Accepted
[Tue May 28 21:05:06 2024] [info] Matched route "api_login_check".
[Tue May 28 21:05:06 2024] [debug] Checking for authenticator support.
[Tue May 28 21:05:06 2024] [debug] Checking support on authenticator.
[Tue May 28 21:05:06 2024] [debug] Notified event "Symfony\Component\Security\Http\Event\CheckPassportEvent" to listener "Symfony\Component\Security\Http\EventListener\UserProviderListener::checkPassport".
[Tue May 28 21:05:06 2024] [debug] Notified event "Symfony\Component\Security\Http\Event\CheckPassportEvent" to listener "Symfony\Component\Security\Http\EventListener\CsrfProtectionListener::checkPassport".
[Tue May 28 21:05:06 2024] [debug] Notified event "Symfony\Component\Security\Http\Event\CheckPassportEvent" to listener "Symfony\Component\Security\Http\EventListener\UserCheckerListener::preCheckCredentials".
[Tue May 28 21:05:06 2024] [info] Authenticator failed.
[Tue May 28 21:05:06 2024] [debug] Notified event "lexik_jwt_authentication.on_authentication_failure" to listener "App\EventListener\UsersJWTResponsesListener::onAuthenticationFailureResponse".
[Tue May 28 21:05:06 2024] [debug] The "Symfony\Component\Security\Http\Authenticator\JsonLoginAuthenticator" authenticator set the failure response.
[Tue May 28 21:05:06 2024] [debug] The "Symfony\Component\Security\Http\Authenticator\JsonLoginAuthenticator" authenticator set the response. Any later authenticator will not be called
[Tue May 28 21:05:06 2024] [debug] Notified event "kernel.request" to listener "Symfony\Component\HttpKernel\EventListener\DebugHandlersListener::configure".
[Tue May 28 21:05:06 2024] [debug] Notified event "kernel.request" to listener "Symfony\Component\HttpKernel\EventListener\Vali

F.3. Déploiement Base de données

```

serveur-backend
/home/eric/Documents/SIMPION/tranquillo/serveur-backend

adminer_tranquill...
adminer:4.8.1
Running
5088:8080 ⓘ
2024-05-28 21:06:22 adminer_tranquillo_4.8.1 | [Tue May 28 19:06:22 2024] PHP 7.4.33 Development Server (http://[::]:8080) started
2024-05-28 21:06:20 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:20+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 11.3.2 started.
2024-05-28 21:06:21 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:21+00:00 [Warn] [Entrypoint]: /sys/fs/cgroup//memory.pressure not writable, functi
B
2024-05-28 21:06:21 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:21+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:21+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 11.3.2 started.
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22+00:00 [Note] [Entrypoint]: MariaDB upgrade not required
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] Starting MariaDB 11.3.2-MariaDB-1:11.3.2+maria~ubu2204 source revision
bf63c33ee42 as process 1
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Compressed tables use zlib 1.2.11
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Using transactional memory
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Number of transaction pools: 1
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Using crc32 + pclmulqdq instructions
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] mariadb: O_TMPFILE is not supported on /tmp (disabling future attempts)
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Using liburib
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Initializing buffer pool, total size = 128.000MiB, chunk size =
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Completed initialization of buffer pool
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Buffered log writes (block size=512 bytes)
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: End of log at LSN=1110748
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Opened 3 undo tablespaces
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: 128 rollback segments in 3 undo tablespaces are active.
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Setting file './ibtmp1' size to 12.000MiB. Physically writing !
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: File './ibtmp1' size is now 12.000MiB.
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: log sequence number 1110748; transaction id 1031
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] Plugin 'FEEDBACK' is disabled.
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] Plugin 'wsrep-provider' is disabled.
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] InnoDB: Buffer pool(s) load completed at 240528 19:06:22
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] Server socket created on IP: '0.0.0.0'.
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] Server socket created on IP: '::'.
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] mariadb: Event Scheduler: Loaded 0 events
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | 2024-05-28 19:06:22 0 [Note] mariadb: ready for connections.
2024-05-28 21:06:22 mariadb_tranquillo_11.3.2 | Version: '11.3.2-MariaDB-1:11.3.2+maria~ubu2204' socket: '/run/mysqld/mysqld.sock' port: 3306 mar
time="2024/05/28 19:06:21" level=Info msg="[smtpd] enabling any authentication (insecure)"
2024-05-28 21:06:21 MAILPIT_tranquillo | time="2024/05/28 19:06:21" level=Info msg="[http] starting on [::]:8025"
2024-05-28 21:06:21 MAILPIT_tranquillo | time="2024/05/28 19:06:21" level=Info msg="[http] accessible via http://localhost:8025/"
2024-05-28 21:06:21 MAILPIT_tranquillo | time="2024/05/28 19:06:21" level=Info msg="[smtpd] starting on [::]:1025 (no encryption)"
2024-05-28 21:06:22 phpmyadmin_in_tranquillo_5.2.1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.2
irective globally to suppress this message
2024-05-28 21:06:22 phpmyadmin_in_tranquillo_5.2.1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.2
irective globally to suppress this message
2024-05-28 21:06:22 phpmyadmin_in_tranquillo_5.2.1 | [Tue May 28 19:06:22.441191 2024] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.59 (Debian) PHP/5
normal operations
2024-05-28 21:06:22 phpmyadmin_in_tranquillo_5.2.1 | [Tue May 28 19:06:22.441485 2024] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'

```

Définitions

• Qu'est-ce qu'un n-uplet ?

Définition :

Un n-uplet, en informatique et en mathématiques, est une séquence ordonnée de n éléments. Le terme "n-uplet" est une généralisation des termes "doublet" (pour deux éléments), "triplet" (pour trois éléments), "quadruplet" (pour quatre éléments), etc. Le "n" représente le nombre d'éléments dans la séquence.

Dans le contexte des bases de données relationnelles :

Relation : Une relation dans une base de données relationnelle est représentée par une table.

Tuple : Chaque ligne de cette table est appelée un tuple ou un n-uplet, où n est le nombre de colonnes de la table.

Colonnes : Les colonnes de la table représentent les attributs ou propriétés de l'entité modélisée par la table.

Exemple :

Considérons une table "Utilisateurs" avec les colonnes suivantes : ID, Nom, Prénom, Email, DateNaissance.

Chaque ligne de cette table, par exemple (1, "Dupont", "Jean", "jean.dupont@example.com", "1990-01-01"), est un 5-uplet, car elle contient 5 éléments.

Les n-uplets sont essentiels dans les bases de données relationnelles car ils permettent de structurer et organiser les données de manière cohérente et normalisée. Chaque n-uplet (ligne) dans une relation (table) représente une instance de l'entité modélisée.

Donc, un n-uplet est une séquence ordonnée de n éléments, et dans le contexte des bases de données relationnelles, il représente une ligne d'une table, où chaque élément correspond à une valeur d'attribut pour cette ligne.

• Qu'est-ce que la Scalabilité

La scalabilité fait référence à la capacité d'un système, d'une application ou d'une infrastructure à s'adapter et à gérer efficacement une augmentation ou une diminution de la charge de travail ou du nombre d'utilisateurs, tout en maintenant des performances acceptables. En d'autres termes, un système est scalable s'il peut évoluer de manière transparente pour répondre aux besoins croissants sans compromettre ses fonctionnalités ou sa performance.

La scalabilité peut être de deux types principaux :

1. Scalabilité Verticale (ou Scaling Up)

Consiste à augmenter les ressources (comme la puissance de calcul, la mémoire, ou le stockage) d'un seul composant du système pour supporter une charge croissante. Par exemple, ajouter plus de CPU ou de RAM à un serveur.



2. Scalabilité Horizontale (ou Scaling Out)

Implique d'ajouter de nouveaux composants identiques au système pour répartir la charge de travail. Par exemple, ajouter de nouveaux serveurs web à une ferme de serveurs pour répartir le trafic.

La scalabilité est un concept essentiel dans la conception et le développement de systèmes informatiques, en particulier dans les environnements où la demande peut fluctuer de manière significative. Une architecture scalable permet à un système de croître de manière harmonieuse avec l'évolution des besoins de l'entreprise ou des utilisateurs, tout en assurant des performances fiables et une expérience utilisateur satisfaisante.

➤ Le fichier SQL

Explication de certaines commandes qui méritent notre attention

- **SET NAMES utf8;**

Configure le jeu de caractères utilisé pour la session actuelle à utf8. Toutes les données échangées entre le client et le serveur seront encodées en UTF-8, ce qui permet de supporter une large gamme de caractères internationaux.

- **SET time_zone = '+00:00';**

Définit le fuseau horaire de la session actuelle à UTC (temps universel coordonné), représenté par +00:00. Cela garantit que toutes les opérations temporelles (comme les enregistrements de date et heure) sont effectuées en UTC, assurant une cohérence des données temporelles indépendamment du fuseau horaire local des utilisateurs.

- **SET foreign_key_checks = 0;**

Désactive les vérifications des clés étrangères pour la session actuelle. Cela permet d'exécuter des opérations de modification de schéma (comme les insertions massives ou les suppressions de tables) sans vérifier les contraintes de clés étrangères, facilitant ainsi les mises à jour importantes ou les réorganisations de la base de données. *Cependant, il est important de réactiver ces vérifications une fois les modifications terminées pour garantir l'intégrité référentielle.*

- **SET NAMES utf8mb4;**

Configure le jeu de caractères pour la session actuelle à utf8mb4, une version améliorée de UTF-8. Cela permet de stocker des caractères supplémentaires tels que des emojis, des symboles asiatiques anciens, et d'autres caractères spéciaux non supportés par l'UTF-8 standard.

- **ENGINE = InnoDB**

Définit le moteur de stockage à utiliser pour les tables de la base de données. InnoDB est un moteur de stockage transactionnel qui supporte les clés étrangères, les transactions ACID (Atomicité, Cohérence, Isolation, Durabilité) et les verrous au niveau des lignes, offrant des performances élevées et une meilleure intégrité des données.

➤ JWT

Un jeton JWT (JSON Web Token) est utilisé pour authentifier les utilisateurs en permettant au serveur de vérifier l'identité du client à chaque requête sans avoir besoin de stocker des informations de session.

Il utilise des signatures ou des hachages pour vérifier que les données n'ont pas été altérées entre client et API.

Un jeton de rafraîchissement (refresh token) est utilisé pour obtenir un nouveau jeton JWT lorsque le jeton actuel expire, permettant ainsi de maintenir une session d'utilisateur sans nécessiter une nouvelle authentification complète.

```
02_DLL_schema.sql M x
database > sql > 02_DLL_schema.sql > ...
D> Run | New Tab Active Connection
1 SET
2   NAMES utf8;
3 D> Run | New Tab
4 SET
5   time_zone = '+01:00';
6 D> Run | New Tab
7 SET
8   foreign_key_checks = 0;
9 D> Run | New Tab
10 SET
11   NAMES utf8mb4;
12 D> Run | New Tab
13 CREATE DATABASE IF NOT EXISTS tranquillo
14 /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci */;
15 D> Run | New Tab
16 USE tranquillo;
17 cf diagramme : annexe C.4.1
```

Ressources

GRUAU Cyril

- Conception d'une base de données

Dominique NANCI - Bernard ESPINASSE

- INGENIERIE DES SYSTEMES D'INFORMATION : MERISE DEUXIEME GENERATION

Wikipedia

- https://fr.wikipedia.org/wiki/Cas_d'utilisation
- UML, etc ...

Grafikart

- <https://grafikart.fr/formations/apprendre-symfony-7>
- <https://grafikart.fr/formations/securite-application-web>
- <https://grafikart.fr/formations/symfony-tests>

Anne Tassot

- Java premier langage

Robin Nixon

- PHP, MySQL JavaScript

Openclassrooms.com

- <https://openclassrooms.com/fr/courses/7709361-construisez-une-api-rest-avec-symfony>
- <https://openclassrooms.com/fr/courses/4087076-testez-fonctionnellement-votre-application-php-symfony/>
- <https://openclassrooms.com/fr/courses/6106191-installez-votre-environnement-de-developpement-java-avec-eclipse>
- ...

Adobe

- <https://experienceleague.adobe.com/fr/docs/contributor/contributor-guide/writing-essentials/markdown>

Axel SHAÏTA

- <https://www.codeheroes.fr/2022/03/15/docker-conteneuriser-son-application/>
- <https://www.codeheroes.fr/2021/10/25/git-pourquoi-ecrire-des-commits-atomiques/>
- <https://www.codeheroes.fr/2023/05/05/typescript-programmation-de-types/>

ChatGPT

- version 3.5
- version 4.0

et aux dizaines voire centaines d'autres ...

Summary of the Tranquillo Organizer Project

The Tranquillo Organizer application is designed to help people, including those with Attention Deficit Hyperactivity Disorder (ADHD) or Autism Spectrum Disorder (ASD), better organize their daily lives. It offers features such as creating flexible reminders, visualizing deadlines, customization options like night mode, and a tracking journal.

The application is specifically intended for people with special needs in time management and organization, particularly those with ADHD and ASD.

For the API development, I used PHP with the Symfony framework, which relies on the Doctrine ORM for database management. The database is MariaDB, deployed locally using Docker.

For the user interface, I developed with NativeScript, which facilitates the development of true native mobile applications using a single code base in JavaScript or TypeScript. Additionally, I used Svelte Native to create a high-performance and responsive mobile application, leveraging optimized JavaScript code to interact directly with native components.

The API will be deployed on a server via a secure SSH connection, and it will be accessible at the secure address '<https://tranquillo.corbisier.fr>'. The database will only be accessible through the API and my user account with the domain and server provider.

The mobile application will be deployed on both Android and Apple platforms.

The application is structured into three main layers:

1. Controllers manage HTTP requests and orchestrate calls to business services;
2. Services contain the business logic and complex operations;
3. Repositories handle data access via Doctrine ORM.

Security is paramount, with the use of JWT for authentication, refresh tokens, and mechanisms to prevent CSRF and XSS attacks. Password hashing ensures their protection.

A detailed testing plan has been developed to verify registration, login, task management, notifications, performance, and security. The tests ensure that the application functions correctly and meets user needs.

Tranquillo Organizer is a mobile application intended to be robust and secure, designed to help people with ADHD and ASD organize their daily lives. The development, deployment, and testing ensure that the application provides a smooth and secure user experience.



Remerciements

Je tiens à exprimer ma gratitude à **Célia DULAC**, notre référente formation, pour son soutien constant et sa présence rassurante tout au long de notre parcours.

Un grand merci à **Hugo ARRU**, notre formateur référent, pour son expertise et ses techniques qu'il a généreusement partagées avec nous.

Frank MARSHALL, notre autre formateur, mérite une mention spéciale pour sa patience et son dévouement, nous guidant avec courage jusqu'à l'examen.

Je remercie également **Simplon**, notre centre de formation, pour l'accueil chaleureux et les conditions optimales qu'ils nous ont offertes.

Un merci sincère à **DSI**, notre employeur, pour la confiance qu'ils ont placée en nous, et tout particulièrement à **Xavier BARROIS** et **Laura CUSENIER** pour leur accompagnement précieux durant ces neuf mois.

Je ne pourrais pas être serein sans exprimer ma profonde gratitude envers tous mes compagnons d'armes. Votre soutien, votre camaraderie et votre détermination ont été essentiels tout au long de ce parcours. Ensemble, nous avons surmonté les défis et partagé des moments inoubliables. Merci à chacun de vous pour votre esprit d'équipe et votre solidarité.

Et enfin, je tiens à remercier **le jury** pour le temps et l'attention qu'ils consacrent à nous évaluer. Votre engagement est grandement apprécié. Merci à tous pour votre soutien et vos encouragements.



THANK YOU!