



MEMOIRE TITRE DE DEVELOPPEUR : 2018 – 2019

CHRISTOPHER MILAZZO



Glossaire

- **FAL** : Final Assembly Line (ligne d'assemblage final d'un avion).
- **C** : Conformité
- **NC** : Non-conformité
- **NCM** : Non-conformité Majeure.
- **JS** : JavaScript.
- **BDD** : Base de données.
- **IHM** : Interface Humain Machine.
- **DAO** : Data Access Object (Objet responsable de l'accès des données en BDD).

Sommaire

1.	Remerciements	6
2.	Abstract	7
3.	Présentation du projet	8
3.1	Présentation de l'entreprise.....	8
3.2	Analyse de l'existant.....	8
3.3	Compétences du référentiel couvertes par le projet.....	9
4.	Cahier des charges.....	10
4.1	Les objectifs de l'application	10
4.2	Périmètre du projet.....	10
4.3	Le type d'application	10
4.4	Les utilisateurs.....	11
4.5	Charte graphique.....	11
5.	Spécifications fonctionnelles.....	12
5.1	Cas d'utilisation	12
5.2	Schéma fonctionnel.....	13
5.3	Maquettage	14
5.4	L'arborescence des vues.....	15
5.5	Présentation des vues	16
5.6	Diagrammes de séquence	22
5.6.1.	Envoi d'une photo – scénario nominal.....	22
5.6.2.	Envoi d'une photo – scénario d'erreur.....	23
5.6.3.	Envoi d'une visite.....	23
5.7	Diagramme de classe.....	24
5.8	Diagramme de déploiement.....	25
6.	Conception	26
6.1	Modèle Conceptuel des Données	26
6.2	Modèle Logique des Données	27
7.	Spécifications techniques.....	28
7.1	Contraintes techniques	28
7.2	Le choix des technologies.....	28
7.3	Outils utilisés	29
8.	Réalisation	31
8.1	Arborescence du projet.....	31

8.1.1.	Application mobile non compilée.....	31
8.1.2.	Application mobile compilée.....	32
8.1.3.	Organisation des classes d'une application type Electron	33
8.1.4.	L'instance J2EE.....	35
8.2	Un objet de gestion d'un élément de liste	36
8.3	La fonction de synchronisation des listes.....	37
8.4	La fonction de la télé-mise à jour.	39
8.5	Pattern MVC	42
8.6	Pattern DAO.....	42
8.9	Pattern singleton	42
8.10	Module pattern	43
9.	Gestion de projet.....	44
9.1	Définition des objectifs et jalons.....	44
9.2	Phases de test.....	44
9.3	Planning	45
10.	Retour d'expérience	46
11.	Conclusion	47
12.	ANNEXES.....	48
12.1	Annexe 1.....	48
12.2	Annexe 2.....	50
12.3	Annexe 3.....	52

1. Remerciements

Je tiens à remercier mes professeurs de l'Adrar pour tous leurs conseils quant au respect des standards ainsi que pour le maquettage.

Merci également à ma compagne d'avoir eu la patience de s'occuper de notre petite fille ainsi que du logis tous les jours week-ends inclus, afin de me libérer du temps pour étudier dans les meilleures conditions possibles.

Merci à Trigo Qualitaire et à mon tuteur de stage de m'avoir laissé ma chance de prouver ce dont j'étais capable.

Et merci à l'Adrar pour cette année formidable et intéressante. Pour la BattleDev de 2018 organisé avec fun, pour la mini borne d'arcade gagnée lors de la BattleDev. Et pour le super barbecue, qui a permis de mieux connaître les autres sessions.

Et pour finir merci au Fongecif Occitanie de m'avoir aidé à financer ma reconversion professionnelle.

2. Abstract

My name is Christopher MILAZZO, I am 31.

I am a computer-holic, this hobby started when I was a kid while I was 8 and playing video games on my game console (especially Street Fighter 2); I was wondering how creators could transform electricity into images that were displayed on television. Or how they could know what button I would press on the controller.

A few years later, I was 13, a neighbor was using Visual Basic 6.0 as entertainment. That marked the beginning of my coding adventure.

Internet was developing and getting more popular. Then, I found the first articles on programming and bought a book dealing with how to learn code in Visual Basic.

A year later I developed a lot of small projects just to practice, then started C++ and DarkBasic and began creating video games.

Last year, I have realized that I could do something else than what I was doing (Fireman) and start discovering some other horizons. That's why I have decided to change jobs.

As part of my training, I met my client that is Trigo Qualitaire. It is a company that offers global quality solutions and is specialized in aeronautic.

My project consists in designing a mobile app (client-server) and client portal that supplies Trigo Qualitaire operator to audit Airbus final assembly lines as well as make sure workers work safely.

The client portal displays all visits and dynamically generates charts and tables with data. I as well as my tutor are very glad with the result. They have even asked to add more functionalities.

3. Présentation du projet

3.1 Présentation de l'entreprise

Le groupe TRIGO a été fondé en 1997. Il est aujourd'hui présent mondialement et compte près de 10000 collaborateurs. Il fournit des solutions opérationnelles de gestion de la qualité, et marginalement, de la gestion de la sécurité des personnes. Son siège est à Nanterre.

TRIGO Qualitaire est une branche qui est spécialisée dans l'aéronautique.

3.2 Analyse de l'existant

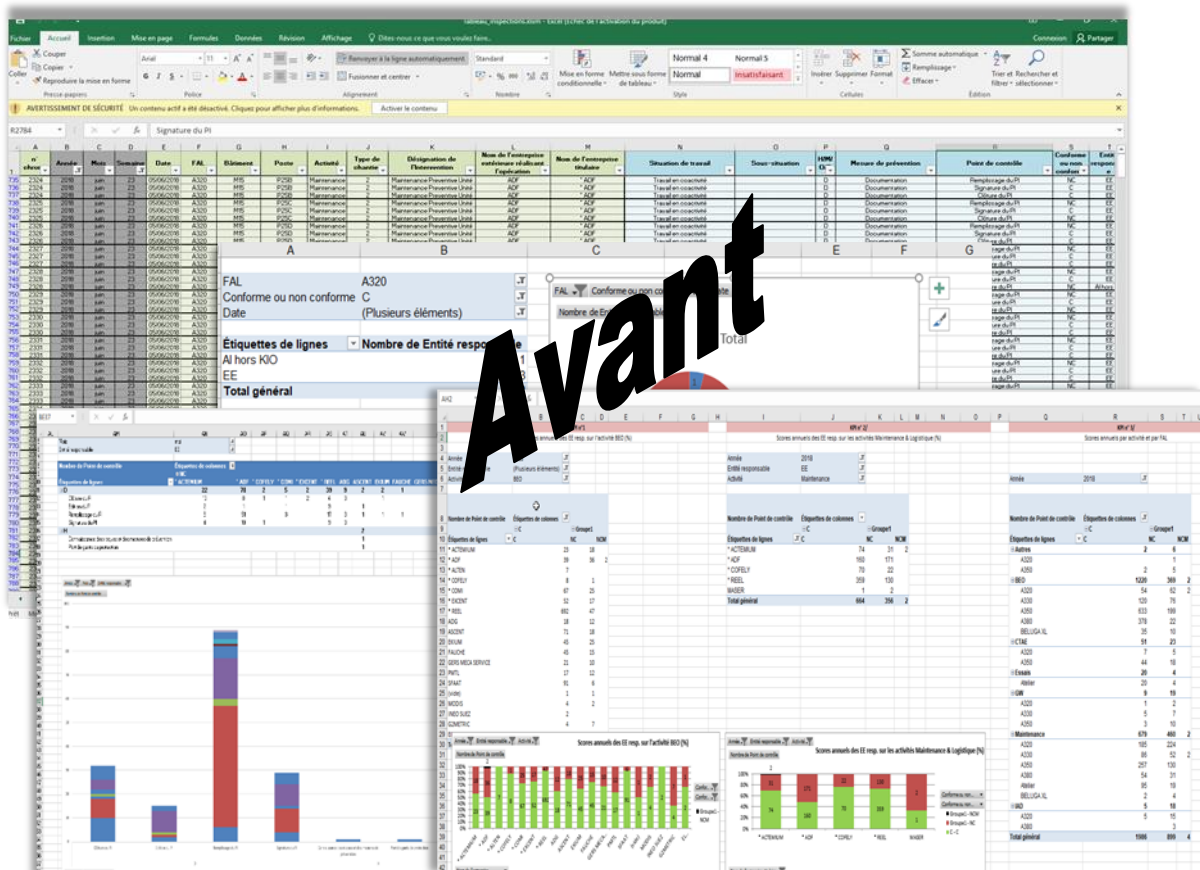
L'application développée concerne l'une des nombreuses prestations réalisées par Trigo-Qualitaire à Airbus. Celle-ci se nomme "Safety guidance" (Pilotage Chantiers). L'ingénieur sécurité placé sur cette prestation a pour rôle d'auditer les FALs (Final Assembly Line) afin de s'assurer que les entreprises sous-traitantes, travaillant sur les postes dans le domaine des moyens industriels, le font en toute sécurité.

Un audit (appelé également visite), s'effectue en vérifiant plusieurs points de contrôle, correspondant chacun à une situation de travail donnée.

Sans l'application, l'ingénieur sécurité utilise des tableaux Excel afin d'enregistrer et d'organiser ses visites, puis doit générer beaucoup de graphes afin de réaliser des indicateurs (mensuels, trimestriels, semestriels et annuels).

Excel n'offrant pas l'ergonomie nécessaire pour une utilisation "mobile", cela force l'ingénieur sécurité à noter ses visites sur un cahier puis l'oblige à retourner au bureau afin de les entrer sur son fichier Excel.

Il y a également un risque d'erreurs dues à la saisie manuelle.



3.3 Compétences du référentiel couvertes par le projet

Ci-dessous la liste des compétences que doit couvrir l'application afin d'être valide pour le Titre de développeur d'application WEB et Mobile :

pour l'activité « Développer une application client-serveur » :

- Maquetter une application
- Concevoir une base de données
- Mettre en place une base de données
- Développer une interface utilisateur
- Développer des composants d'accès aux données

pour l'activité « Développer une application Web »

- Développer des pages web en lien avec une base de données
- Mettre en œuvre une solution de gestion de contenu ou d'e-commerce
- Développer une application simple de mobilité numérique
- Utiliser l'anglais dans son activité professionnelle en informatique

4. Cahier des charges

4.1 Les objectifs de l'application

Le besoin exprimé par Trigo-Qualitaire est non seulement un besoin d'amélioration du quotidien de son salarié (l'ingénieur sécurité) en limitant les tâches répétitives et fastidieuses, mais aussi un besoin de gain de temps afin qu'il puisse délivrer plus de formations préventives aux entreprises auditées.

Concernant le client de Trigo-Qualitaire (Airbus), c'est le besoin d'accès rapide aux données brutes et aux indicateurs sans temps de latence qui a été exprimé. Ce besoin sera comblé par le biais d'un portail WEB qui mettra à disposition d'Airbus une page de consultation de visites munie de multiples filtres, ainsi que d'une page capable de générer les indicateurs en temps réel et de manière entièrement automatisée.

4.2 Périmètre du projet

Il est prévu que l'application soit utilisée sur la mission Pilotage Chantiers à Toulouse. Mais il n'est pas exclu qu'elle puisse être aussi utilisée plus tard sur une autre prestation du groupe Trigo.

C'est donc dans cette hypothèse et pour adopter de bonnes habitudes que j'ai commenté et documenté mon code source entièrement en Anglais.

4.3 Le type d'application

C'est une application de type client-serveur qui se classe dans la catégorie utilitaire, mettant à disposition une interface graphique pour saisir et sauvegarder des données le plus efficacement possible.

Plus précisément, l'opérateur effectue une visite et saisie sur l'application tous les points de contrôles qu'il effectue lors de la visite.

Pour chaque point de contrôle effectué, l'opérateur coche **C**, **NC** ou **NCM** en fonction de la situation : « Conformité », « Non-Conformité » ou « Non-Conformité Majeure » respectivement.

Il ajoutera une entité responsable de cette conformité/non-conformité, ainsi qu'un commentaire s'il le désire.

Pour chaque NC et NCM, une action est définie afin que ces dernières cessent et/ou ne se reproduisent pas.

L'application permet également d'associer des photos à une visite. Enfin elle propose une interface regroupant des listes servant de rappel (actions non définies, actions non closes sur le point d'avoir la date limite dépassée, ETC...).

Concernant les utilisateurs Airbus, il leur est mis à disposition un portail d'accès. Il s'agit de la même application que pour l'opérateur, mais en version navigateur et tronquée afin de supprimer les fonctionnalités réservées à l'opérateur.

Plus bas, (chapitre 5.5) des présentations plus exhaustives et plus parlantes des vues sont disponibles.

4.4 Les utilisateurs

Il y a deux types d'utilisateurs pour l'application :

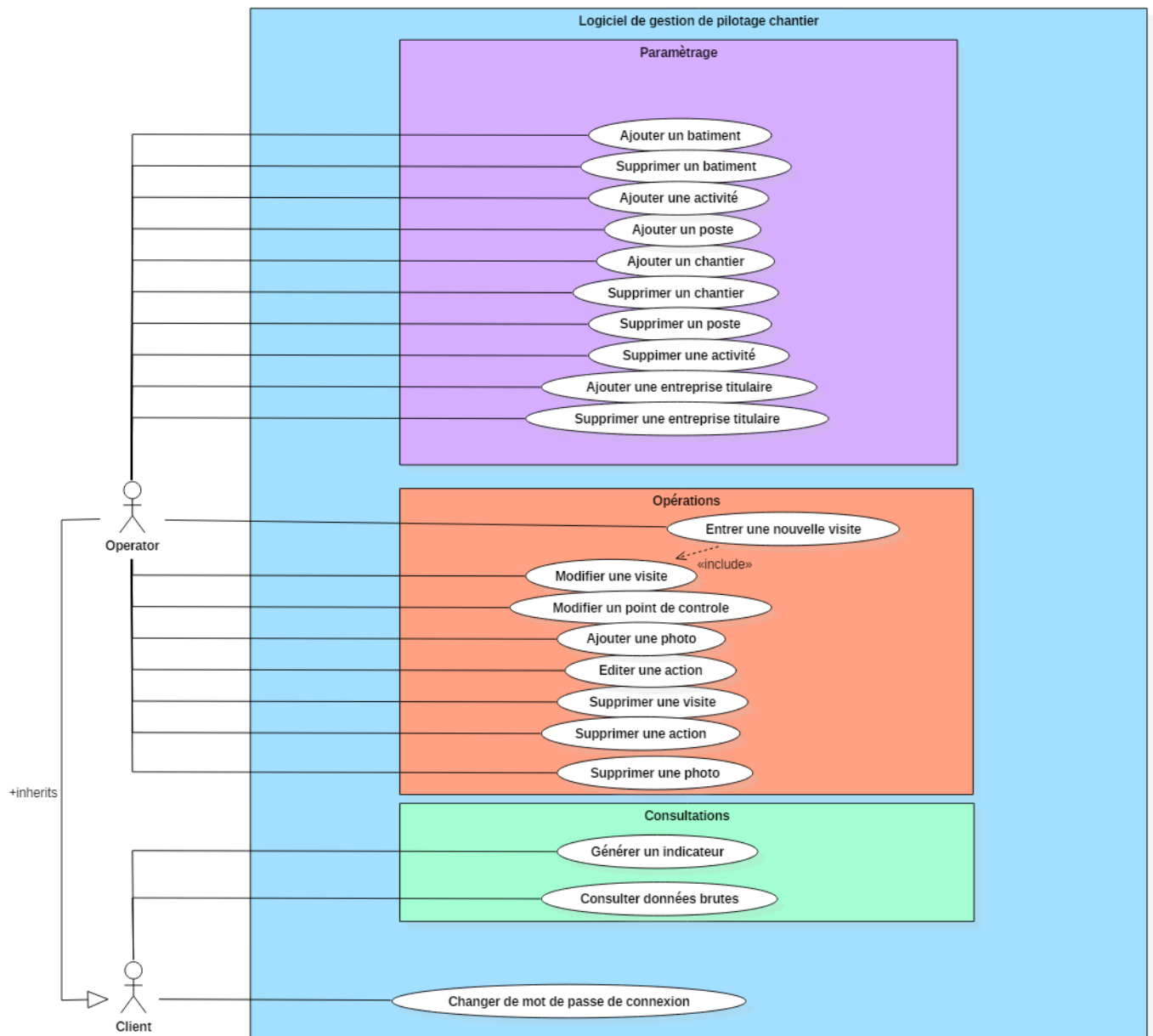
- L'opérateur. Il s'agit de l'ingénieur sécurité Trigo Qualitaire, qui alimentera l'application de données.
- Les superviseurs. Il s'agit des chargés d'affaire Airbus, qui gèrent les chantiers d'installation, de modification et de maintenance des moyens industriels. Ces utilisateurs peuvent consulter à tout moment les données brutes des visites. Puis ils peuvent générer des graphiques (indicateurs) dont ils ont choisi initialement le contenu et la forme. L'objectif final est d'avoir une vue globale sur la sécurité au travail et de "piloter" leurs sous-traitants.

4.5 Charte graphique

Aucune exigence n'a été définie. Mais cela sera fait par la suite si le client le demande. Comme je lui ai expliqué, il n'y aura aucune difficulté à modifier les couleurs et l'organisation des éléments d'une page par le biais des feuilles de style CSS. Néanmoins, j'ai pris le soin de récupérer les codes couleurs utilisés sur le site web du groupe Trigo (<https://www.trigo-group.com/>) afin de pouvoir proposer quelque chose de pertinent dès le début.

5. Spécifications fonctionnelles

5.1 Cas d'utilisation



5.2 Schéma fonctionnel

Application mobile – sur tablette tactile masterisée Airbus, utilisée par l'ingénieur sécurité



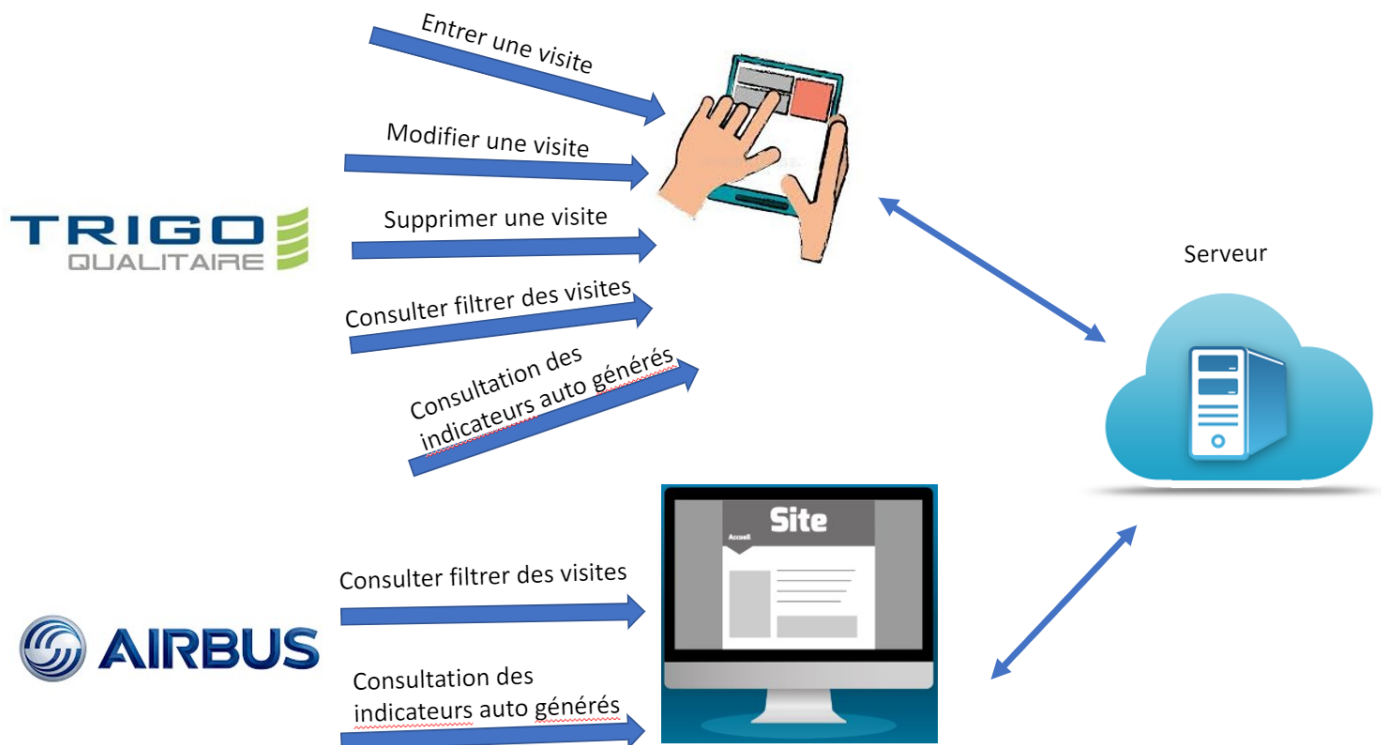
Portail client – Application WEB pour navigateur utilisé par le client et l'ingénieur sécurité.



Application serveur – sur VM Serveur Windows 2016



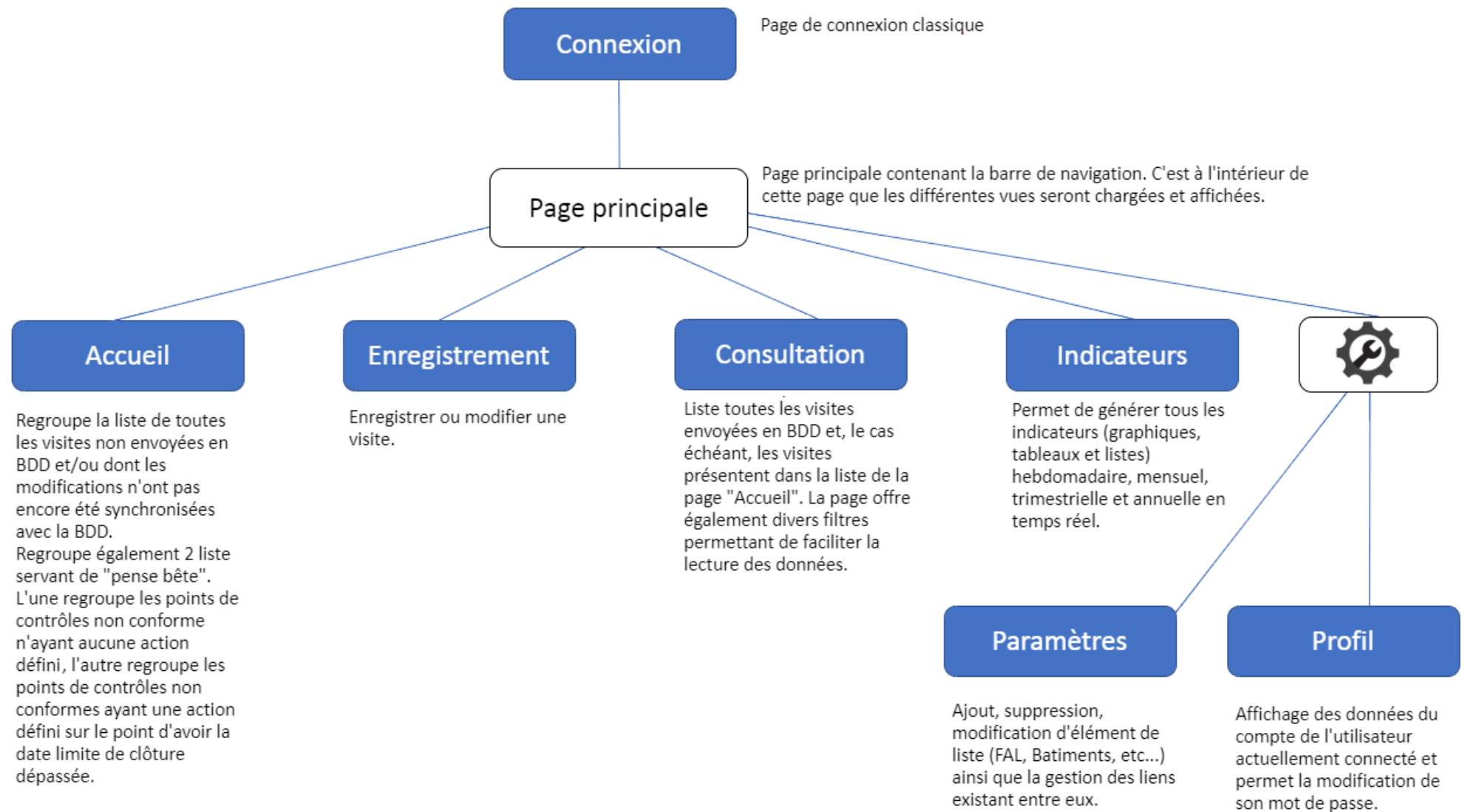
SGBD : PostgreSQL v10.4



5.3 Maquettage

Les interfaces ont été réalisées et pensées intégralement par moi-même. Certaines ont été améliorées grâce à l'aide et aux conseils précieux de mes professeurs de l'Adrar, que je remercie encore. Elles ont par la suite été vérifiées, testées et approuvées par l'ingénieur sécurité de Trigo Qualitaire.

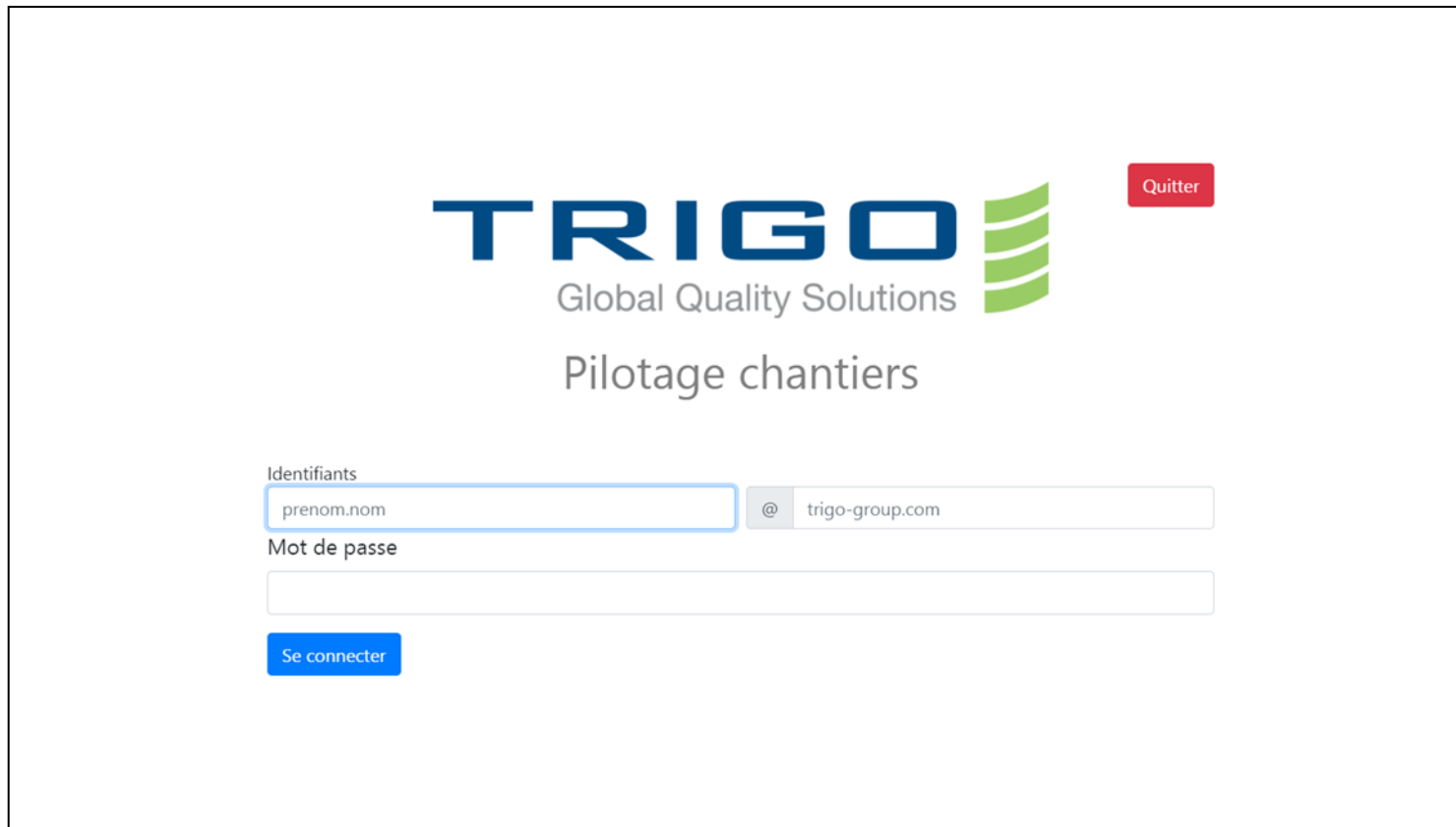
5.4 L'arborescence des vues



5.5 Présentation des vues

Ci-dessous, des captures d'écran présentant quelques pages de l'application mobile (les interfaces pour le portail client seront strictement les mêmes) :

Page de connexion : celle-ci propose également de démarrer en mode hors-ligne, par le biais d'un bouton qui apparaît au bas de la page. Le mode hors ligne est disponible uniquement pour les utilisateurs s'étant connectés au préalable et ne s'étant pas déconnectés depuis.



The screenshot displays the login interface of the TRIGO mobile application. At the top center, the TRIGO logo is shown in blue, followed by the text "Global Quality Solutions" in grey. To the right of the logo is a green icon consisting of three horizontal bars. Below the logo, the text "Pilotage chantiers" is displayed in grey. In the top right corner, there is a red button labeled "Quitter". The login form consists of two input fields: the first is labeled "Identifiants" and contains the text "prenom.nom"; the second is labeled "Mot de passe" and is currently empty. To the right of the "Identifiants" field, there is a grey button with an "@" symbol and the text "trigo-group.com". Below the "Mot de passe" field, there is a blue button labeled "Se connecter".

Page d'accueil (uniquement disponible pour l'opérateur) : permettant d'avoir une vue globale sur les visites n'ayant pas encore été enregistrées en base de données, ainsi que les points de contrôles demandant une attention particulière :

- Les points de contrôles non conformes n'ayant aucune action définie.
- Les points de contrôles ayant une action non close et sur le point d'avoir la date de clôture limite dépassée.

Accueil **Enregistrement** **Consultation** **Indicateurs**

Rien à afficher

proche ou dépassée

20/11/2018 - * [redacted] - Réparation batterie
A330 - C23 - ATELIER

NC - Poste de travail rangé et propre

NC - Dégagement des voies de circulation

27/11/2018 - * [redacted] - NCM A330 - C23 - ATELIER

NCM - Dégagement des voies de circulation

Visites non enregistrées en base

# visit	Date	Désignation visite	FAL	Bâtiment	Poste	Titulaire
---------	------	--------------------	-----	----------	-------	-----------

Deuxième page d'enregistrement d'une visite : regroupe toutes les situations de travail que peut rencontrer l'opérateur durant la visite d'un chantier sur une FAL.

En bas à gauche se trouve une icône d'appareil photo permettant la gestion des photos (ajout, suppression).



Troisième page d'enregistrement : l'opérateur y accède en cliquant sur une situation de travail. Sur la page sont listés tous les points de contrôle correspondant à la situation de travail sélectionnée.

Chaque ligne contient des cases à cocher permettant de définir si le point de contrôle est conforme (l'ouvrier travail en sécurité et respecte les règles), non conforme ou non conforme « majeur ».


Chaque point de contrôle non-conforme ou non-conforme « majeur », comporte un bouton en fin de ligne permettant d'ouvrir un panneau latéral, d'où l'opérateur définit une action corrective à réaliser.

Accueil



Enregistrement
en cours : Ma visite

Consultation

Indicateurs




Points de contrôle pour : Électricité

Point de contrôle	C	NC	NCM	Sous situation	Entité responsable	Commentaire	Action
Détention d'une habilitation électrique adéquate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Aucune ▼		
EPC adaptés et en état	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Aucune ▼		
Etat du capotage, des gaines...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		EE ▼		
Fiche de consignation complétée	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		AI KIO ▼		
Nappe et tapis isolants	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		Aucune ▼	Commentaire	
Outils isolés	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Aucune ▼		
Port de casque / protection faciale	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Aucune ▼		
Port de chaussures isolantes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Aucune ▼		
Port de gants électriques adaptés	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Aucune ▼		
Registre de consignation complété	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Aucune ▼		

Annuler

☐ actions non définies

Valider 

Page consultation : permet la consultation des données.

Comporte :

- Un panneau avec des filtres permettant de rechercher rapidement une ou plusieurs visites.
- La liste des visites.
- Le contenu d'une visite (liste des points de contrôle rassemblés par situation de travail).

Accueil

Enregistrement

Consultation

Indicateurs

Rechercher par

date

De

x

Activité

Batiment

EE Titulaire

FAL

Poste

Désignation de la visite

visit

Type de chantier

1 2

PI

12 visite(s)

Vi...	Date	Semaine	Désignation	FAL	Batiment	Poste	Titulaire
7	02/01/2019	S 1	Modification diable	A350	A50	P30C	
1	02/01/2019	S 1	Projet REACH : mo...	A330	C02	P40B	
3	02/01/2019	S 1					

Atelier

Activité : Maintenance

Type de chantier : 2

Titulaire :

Réalisant : Actemium

PI:0 Electronique

Modifier

Accessibilité des extincteurs et autre matériel de lutte contre l'incendie

EE

Extincteurs dans salle soudure non accessibles. NB : pas de soudure en cours.

Action clôturée

Rendre les extincteurs accessibles

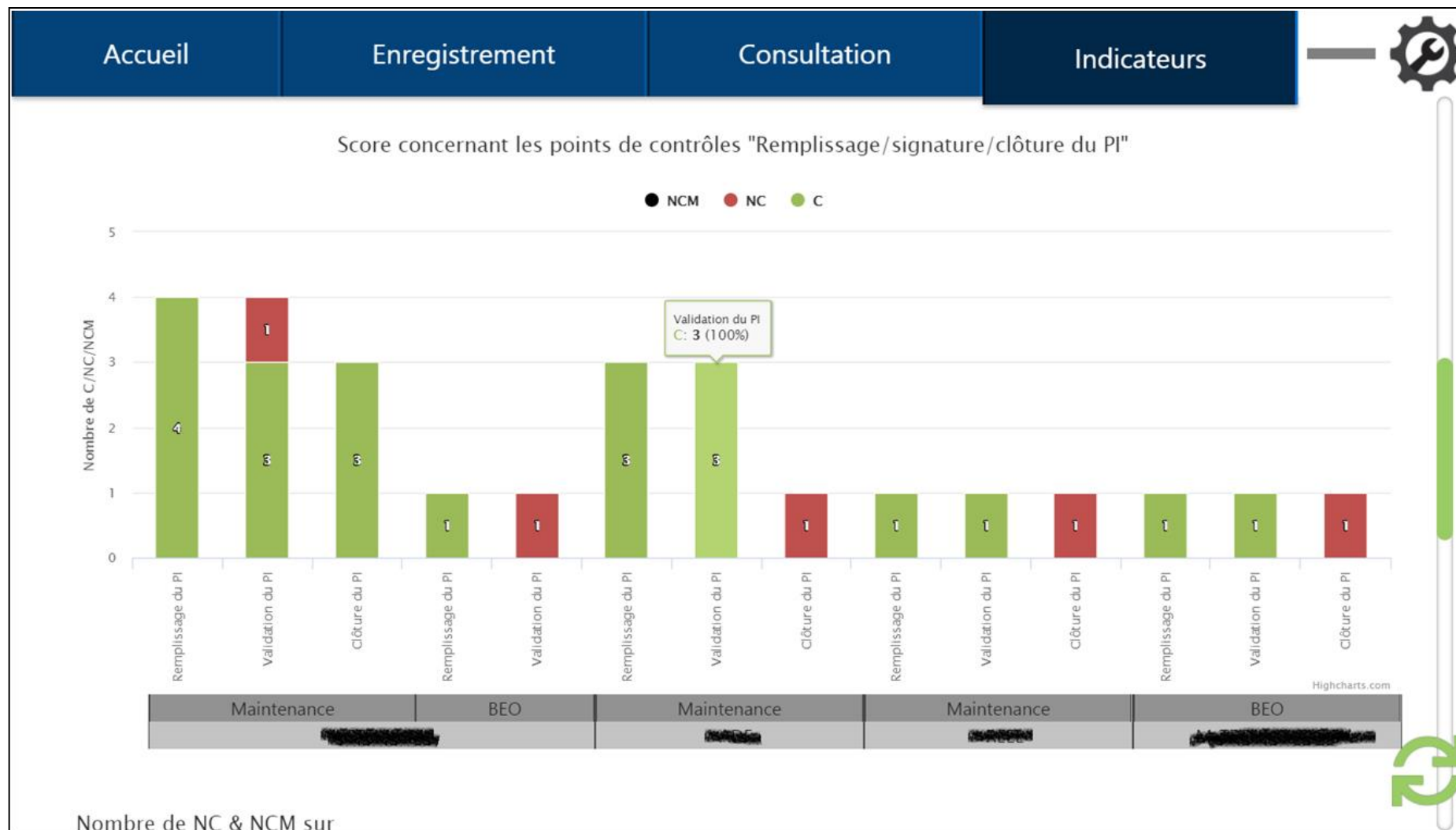
Entité porteuse : EE

Date Limite : 02/01/2019

Date de clôture : 02/01/2019

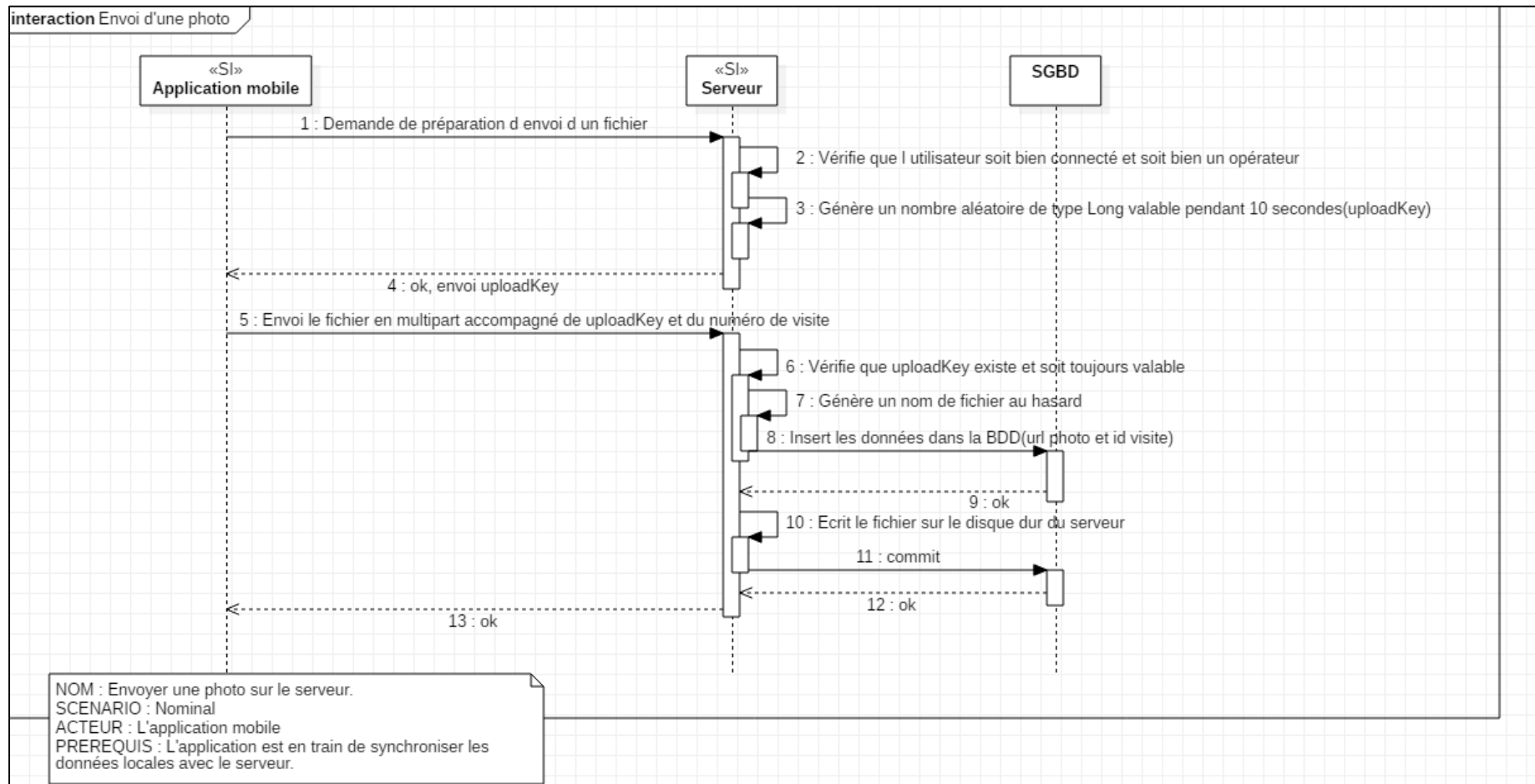
Port des chaussures de sécurité

Page « indicateurs » : regroupe tous les graphiques permettant à Airbus d'effectuer le pilotage des chantiers.



5.6 Diagrammes de séquence

5.6.1. Envoi d'une photo – scénario nominal



5.6.2. Envoi d'une photo – scénario d'erreur

NOM : Envoyer une photo sur le serveur
SCENARIO : Erreur

Ce scénario commence au point 10 du scénario nominal.

10.1 : L'écriture du fichier à échouée.

10.2 : Rollback.

10.3 : Afficher l'origine de l'erreur dans les logs du serveur.

10.4 : Renvoyer un code erreur HTTP 500 à l'application mobile.

FIN DU SCENARIO

5.6.3. Envoi d'une visite

Pour des raisons de place et de lisibilité le diagramme a été placé en annexe 1.

Même s'il peut paraître un peu compliqué de prime abord, j'ai tenu à le mettre dans le mémoire car :

- On peut y voir le mécanisme du pattern factory à l'oeuvre.
- On voit le fonctionnement d'une transaction (3 DAO utilisés pour sauvegarder une visite).

En résumé, l'envoi d'une visite se passe come suit :

- Le front controler reçoit la requête.
- Il appelle le controleur correspondant, qui va récupérer les données (la visite) et la parser. Puis il appelle l'objet métier.
- L'objet métier s'occupe de demander l'ouverture d'une connexion vers la BDD grâce à la DAO factory. Puis il récupère une référence vers les 3 DAO.
- Il enregistre l'entête de la visite avec l'objet PostgresVisitDAO
- Il enregistre tous les points de contrôle de la visite via PostgresControlPointDAO
- Il enregistre, le cas échéant, les actions associées aux points de contrôle.
- Si tout s'est bien passé, l'objet métier COMMIT et ferme la connexion. Puis il renvoie l'ID auto-généré de la visite qui vient d'être enregistrée.

- Le controleur prépare la réponse (http Code 200 + ID visite).
- Le front controler renvoie la réponse à l'application mobile.

NB : Les photos associées aux visites sont envoyées par la suite via une requête à part. C'est pour cela que l'ID visite est envoyé, afin de faire le lien entre les photos envoyées et leur visite associée.

Remarque : J'ai volontairement omis de faire apparaître les deux filtres dans le diagrammes de séquence objet afin d'éviter de l'alourdir inutilement. En effet ils n'apportent rien d'intéressant, ils se contentent de vérifier si l'utilisateur est bien connecté et si le compte utilisateur est bien autorisé à faire ce type de requête.

5.7 Diagramme de classe

Pour une meilleure lisibilité le diagramme est présent en **annexe 2**.

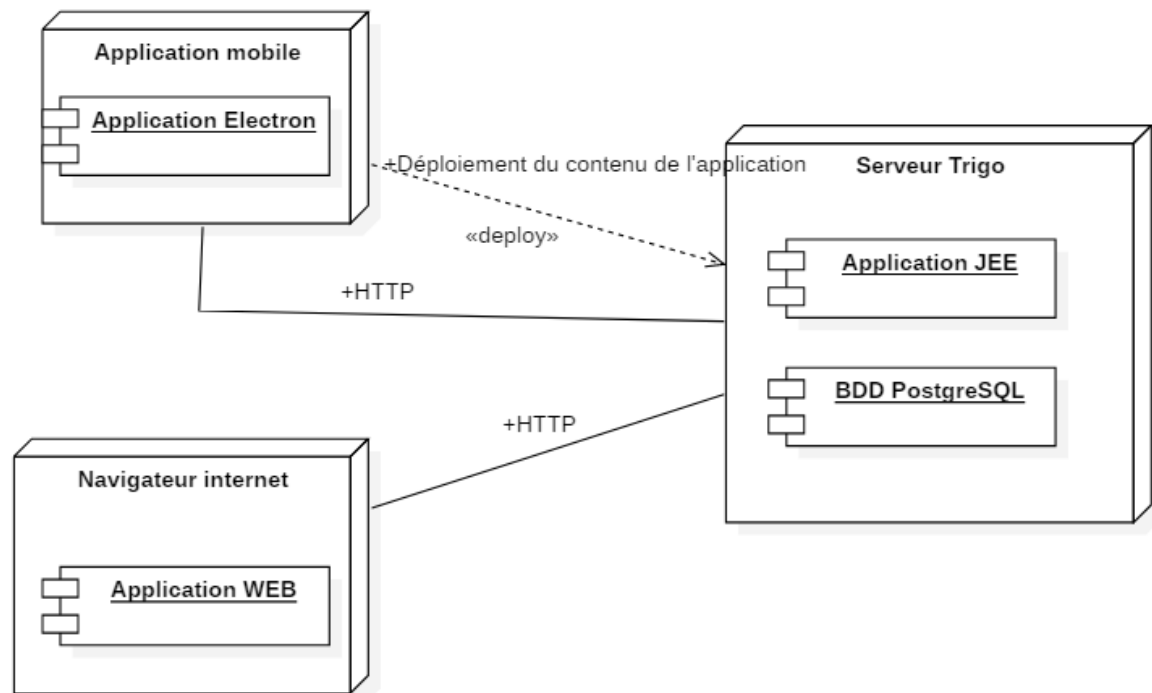
Il est à noter que le diagramme présente une particularité. En effet, les entités blanches représentent des données enregistrées en localStorage.

Pour faire simple : les entités de couleurs représentent mes couches « model » et « controler », donc l'état de mes données à afficher sur l'interface utilisateur et la logique de l'application. Tandis que les entités blanches sont les données enregistrées et persistantes.

Les données seront au maximum synchronisées avec le localStorage, afin d'éviter une perte de données en cas de crash de l'application ou autre soucis comme l'extinction prématurée de la tablette, suite à batterie faible par exemple.

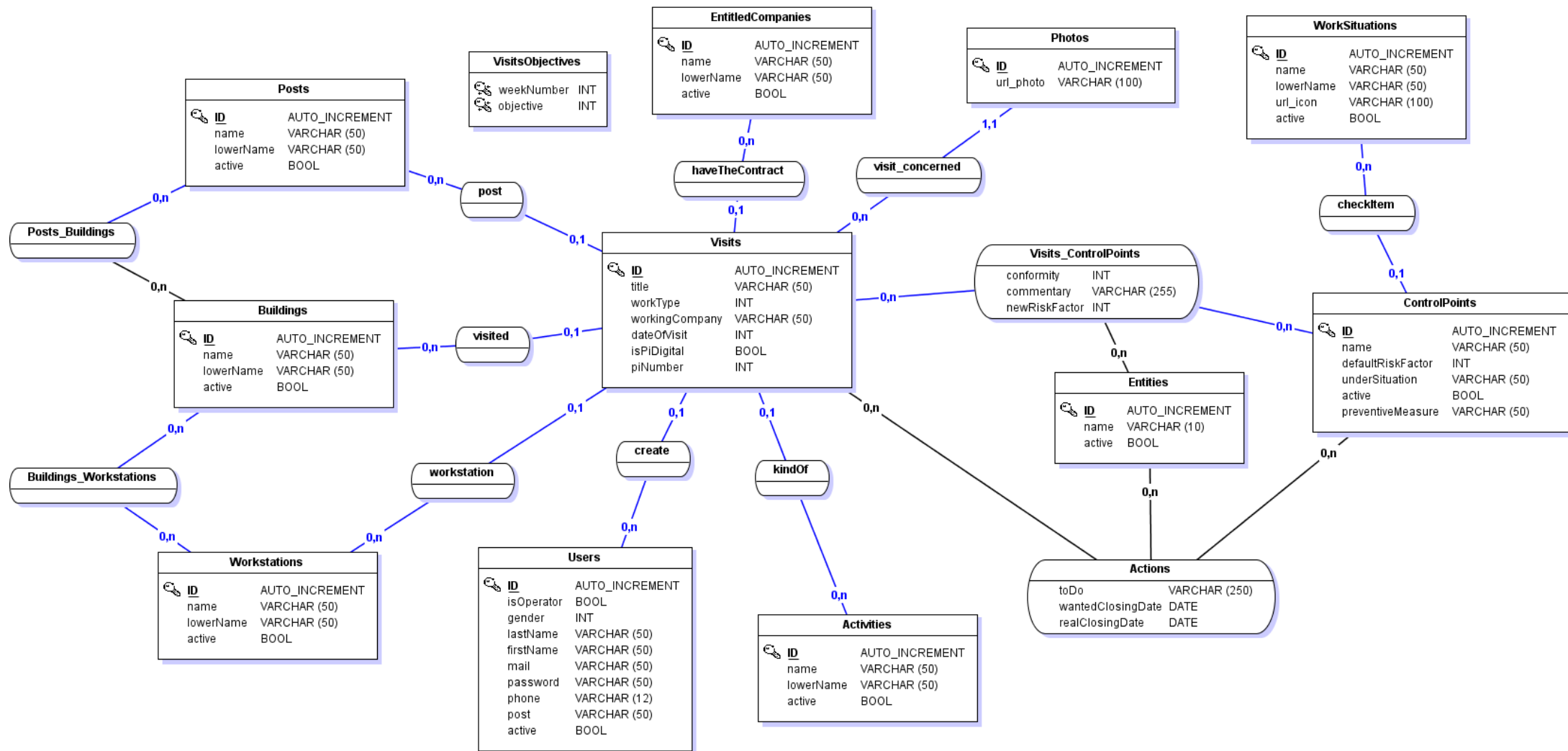
5.8 Diagramme de déploiement

Ci-dessous le diagramme de déploiement, permettant d'avoir une vue d'ensemble sur l'organisation général du projet.

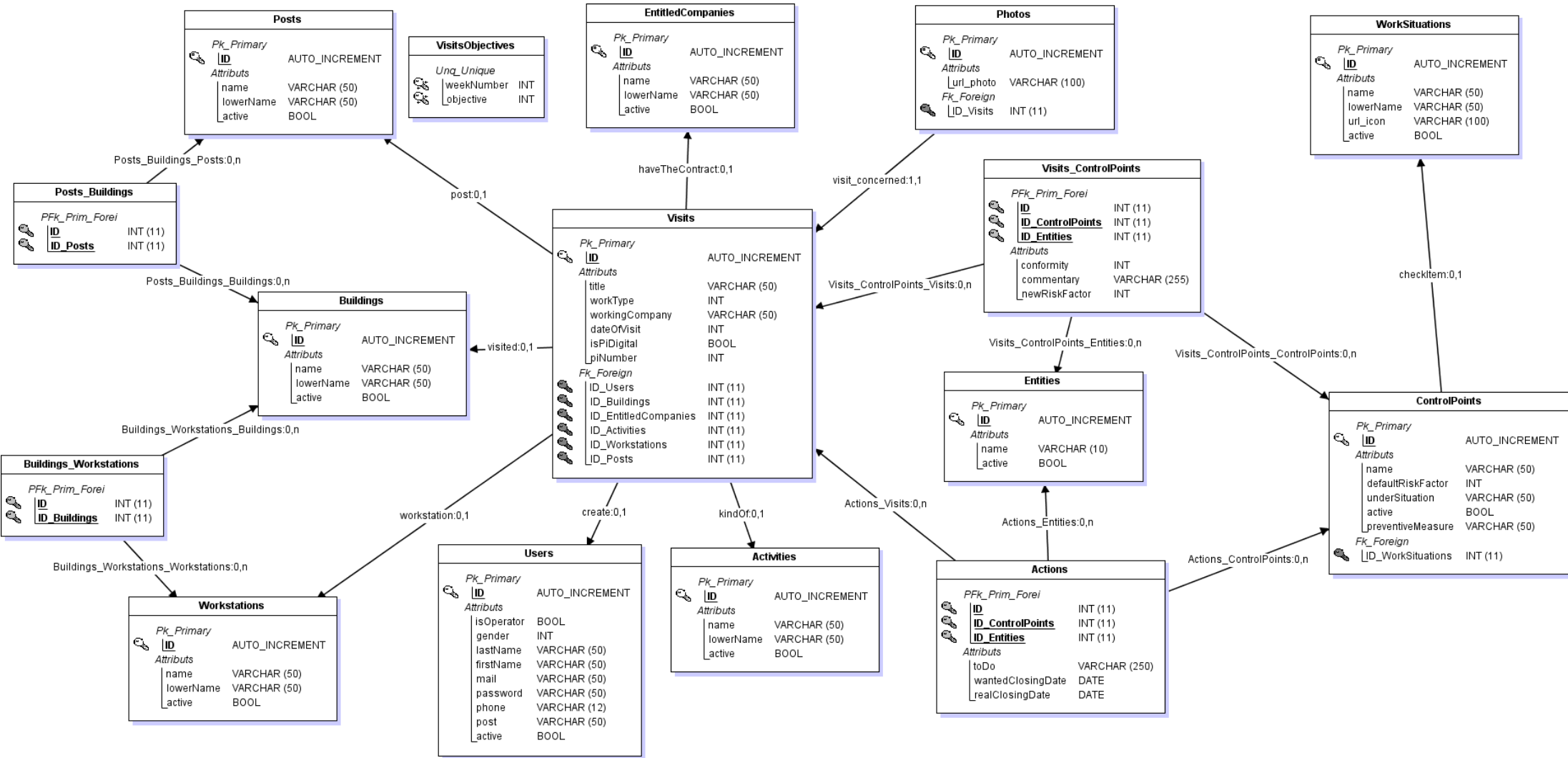


6. Conception

6.1 Modèle Conceptuel des Données



6.2 Modèle Logique des Données



7. Spécifications techniques

7.1 Contraintes techniques

Le seul besoin technique de l'application est un serveur mis à disposition et entretenu par Trigo Qualitaire, sur lequel la base de données PostgreSQL et l'instance J2EE tourneront.

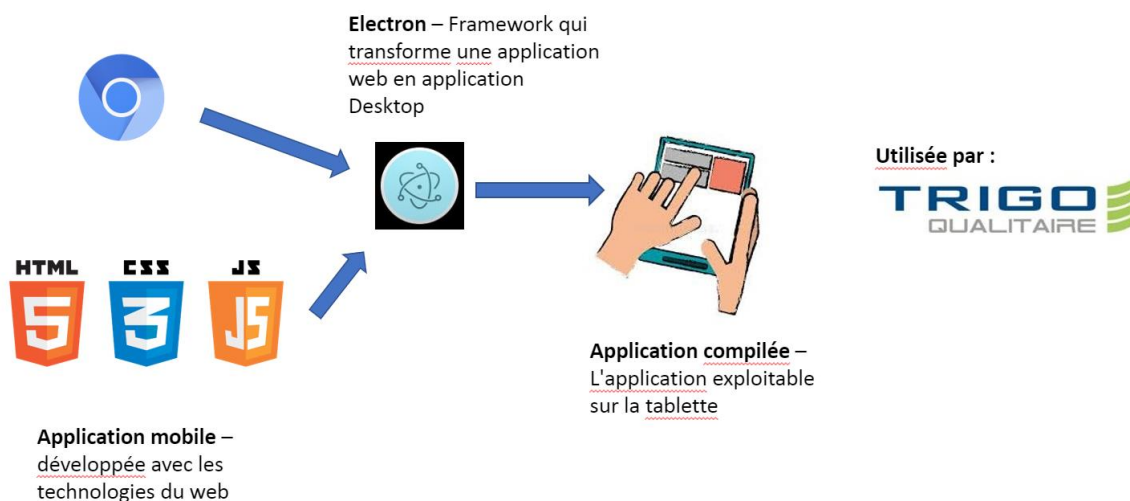
Etant donné que l'application est de petite envergure, aucun Framework n'a été utilisé.

Bootstrap a été utilisé, mais uniquement pour le design des formulaires, des boutons, des barres de progression et pour la grille de positionnement responsive.

7.2 Le choix des technologies

Pour l'application mobile, aucune technologie n'a été imposée. Mon choix s'est donc porté sur les technologies du web. En effet elles permettent de mettre en place simplement des interfaces graphiques esthétiques, et offrent la possibilité de factoriser le code source de l'application mobile et du portail client. Cela divise par 2 le travail.

Moteur web Chromium –
utilisé par Electron



En revanche, pour le serveur, Trigo Qualitaire m'a contraint à utiliser une base de données PostgreSQL et une instance Java EE. Le choix de Trigo Qualitaire est motivé par la volonté d'uniformiser ce projet avec un autre projet déjà en cours de développement.

Portail client – développée avec les technologies du web



7.3 Outils utilisés



Visual Code studio : IDE généraliste développé par Microsoft. Je l'ai utilisé pour le HTML, SCSS, JavaScript et l'écriture des scripts Bash.



Prepros : permet de compiler du Sass et de minimifier le JavaScript.



Electron App : le frameWork permettant de transformer une application Web en une application Desktop pour Linux, Mac et Windows en s'appuyant sur le moteur Chromium.



Script Bash : m'a permis de créer le script d'automatisation de la compilation. En effet, malgré le fait que la compilation soit effectuée par un module node appelé electron-packager, le script m'a évité la tâche rébarbative de devoir renommer tous les fichiers *.min.js en *.js et de devoir paramétrer le 'resources path' et la version de l'application à l'intérieur du fichier de configuration « app.config ».



Eclipse : un IDE extrêmement puissant développé par les membres de « Eclipse Foundation » que j'ai utilisé pour coder l'instance J2EE.



PGAdmin V4 : une interface de gestion du Système de Gestion de Base de Données PostgreSQL. PostgreSQL est un projet open source qui a su s'imposer et qui fait ses preuves depuis 1996.

8. Réalisation

8.1 Arborescence du projet

8.1.1. Application mobile non compilée

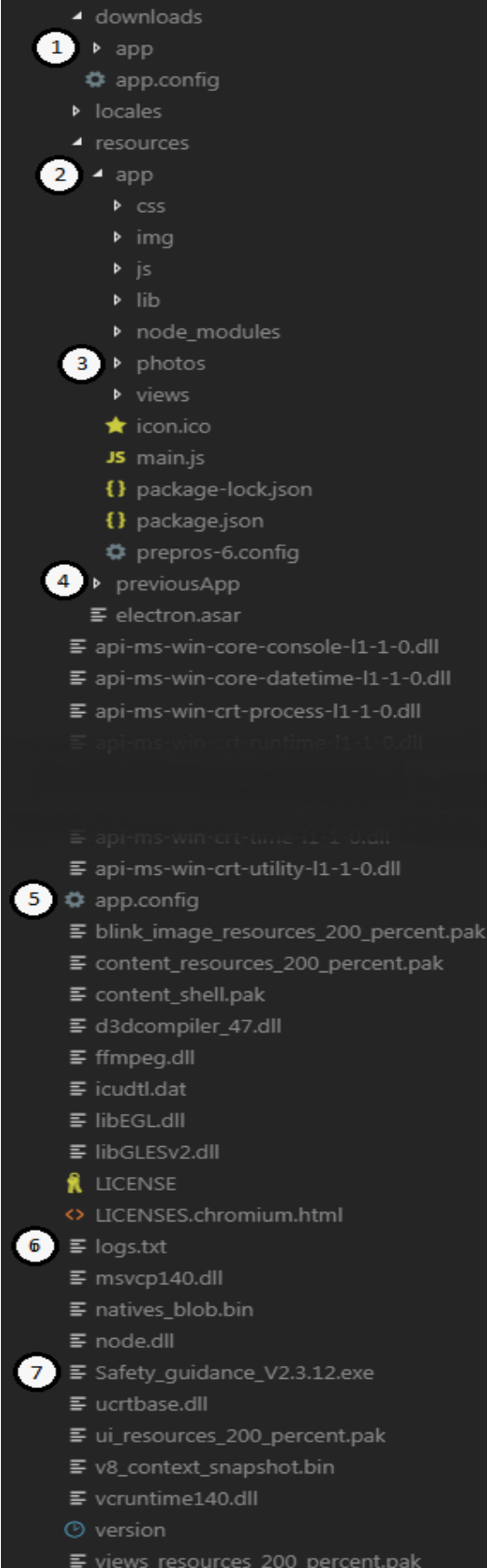
Ci-dessous un bref descriptif de l'organisation du projet coté application mobile :

Voir annexe 3 pour plus de précisions sur les autres dossiers.

<ul style="list-style-type: none">▶ bootstrap-4.1.0▶ css▶ doc▶ img▲ js<ul style="list-style-type: none">▶ Charts▶ controlers▲ items<ul style="list-style-type: none">▶ dynamics▶ statics▶ requiredFirst▶ user▶ visits▶ lib▶ node_modules▶ release-builds▶ scss▶ views⚙ app.config📄 compilation.sh📄 docgenerator.sh★ icon.icoJS main.js{ } package-lock.json{ } package.json⚙ prepros-6.config	<h4>Description du contenu du dossier JS</h4> <ul style="list-style-type: none">• Charts : Objets spécialisés dans la génération des graphiques, tableaux et listes pour les indicateurs.• controlers : Regroupe tous les contrôleurs responsables de la dynamisation et du comportement des vues HTML.• Items<ul style="list-style-type: none">➤ dynamics : Regroupe tous les objets de gestion des éléments de liste dits dynamiques. C'est-à-dire qu'ils sont gérés directement par l'opérateur via une interface de paramétrage.➤ statics : Regroupe tous les objets de gestion des éléments de liste dits statiques. C'est-à-dire qui ne peuvent être gérés uniquement par le billet de la BDD.• requiredFirst : Regroupe tous les objets importants ou qui n'ont pas de catégorie particulière. (DateHandler par exemple).• users : Regroupe tous les objets de gestion des utilisateurs.• visits : Regroupe tous les objets de gestion des visites.
--	--

8.1.2. Application mobile compilée

Cette fois-ci l'application mobile « compilée » avec Electron :

 <ul style="list-style-type: none">1 - downloads<ul style="list-style-type: none">appapp.configlocales2 - resources<ul style="list-style-type: none">app<ul style="list-style-type: none">cssimgjslibnode_modules3 - photosviewsicon.icomain.jspackage-lock.jsonpackage.jsonprepros-6.config4 - previousApp<ul style="list-style-type: none">electron.asarapi-ms-win-core-console-l1-1-0.dllapi-ms-win-core-datetime-l1-1-0.dllapi-ms-win-crt-process-l1-1-0.dllapi-ms-win-crt-runtime-l1-1-0.dllapi-ms-win-crt-utility-l1-1-0.dll5 - app.configblink_image_resources_200_percent.pakcontent_resources_200_percent.pakcontent_shell.pakd3dcompiler_47.dllffmpeg.dllicudtl.datlibEGL.dlllibGLESv2.dllLICENSELICENSES.chromium.html6 - logs.txtmsvcp140.dllnatives_blob.binnode.dll7 - Safety_guidance_V2.3.12.exeucrtbase.dllui_resources_200_percent.pakv8_context_snapshot.binvcruntime140.dllversionviews_resources_200_percent.pak	<p>1 - downloads/app : contient la dernière version en cours de téléchargement.</p> <p>2 - resources/app : contient l'application.</p> <p>3 - resources/app/photos : contient les photos prises et téléchargées pour les visites.</p> <p>4 - resources/previousApp : contient la version précédente (avant la dernière mise à jour).</p> <p>5 - app.config : configuration de l'application (adresse du serveur, version, etc....).</p> <p>6 - logs.txt : les messages générés par l'application durant l'exécution.</p> <p>7 - Safety_Guidance.exe : démarre l'application.</p>
---	---

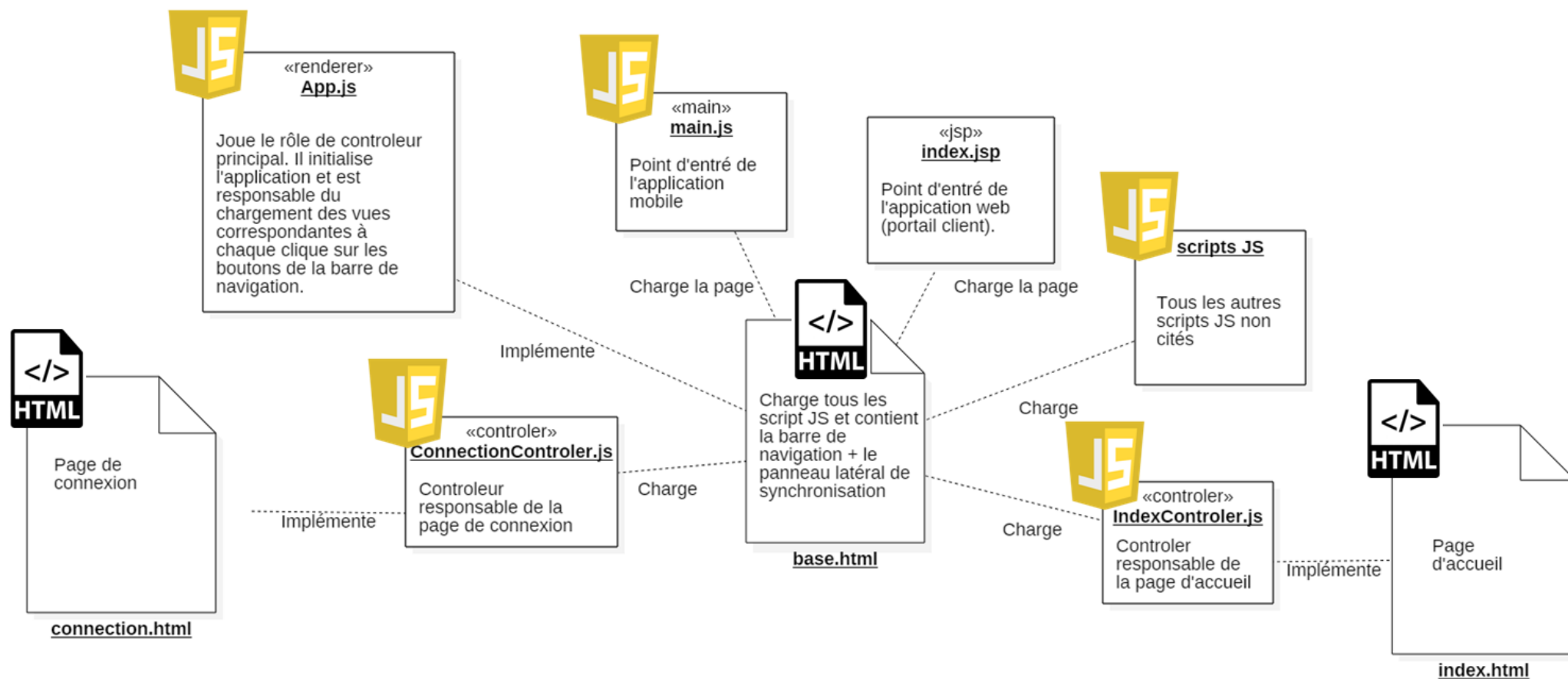
8.1.3. Organisation des classes d'une application type Electron

Le projet mobile est un projet ElectronApp.

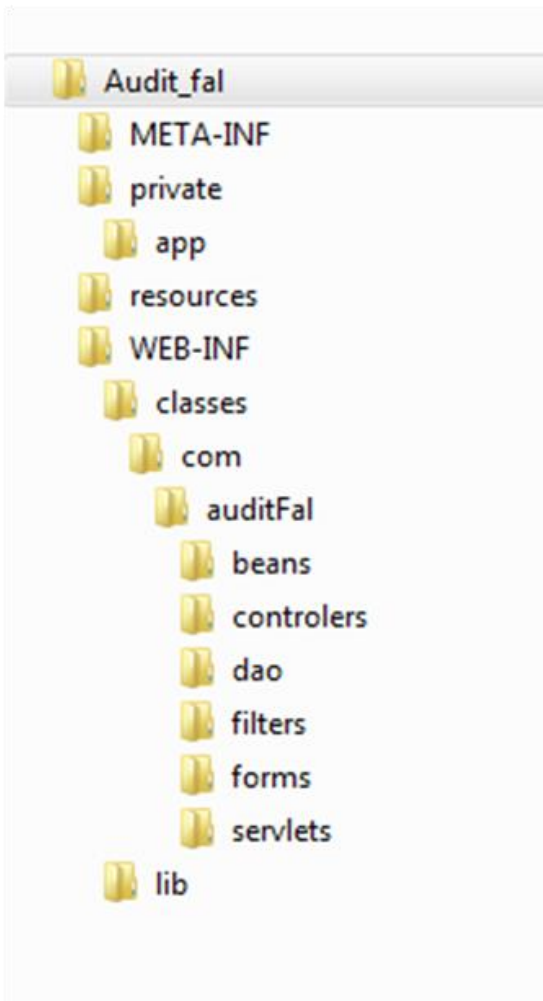
Une application Electron s'articule de la manière suivante :

Un fichier est le point d'entrée de l'application appelé « main process », et tout autour gravitent les fichiers responsables de l'implémentation du comportement des vues. Ceux-là sont appelés « renderer ».

Afin de pouvoir garder la même application entre la version mobile et la version portail client, j'ai décidé d'organiser les classes comme suit :



8.1.4. L'instance J2EE



- **private/app** : le contenu de l'application client
- **resources** : contient le style css et script JS pour le fonctionnement de page JSP (page de connexion pour le portail client).
- **beans** : model de données
- **controlers** : les classes de la couche controler
- **dao** : les classes « Data Access Object »
- **filters** : contient les deux filtres (un pour la connexion l'autre pour les droits des utilisateurs.
- **forms** : Les classes métier responsables de toute la logique de l'application.
- **servlets** : contient les 2 servlets ; l'un gère les requêtes entre le serveur et l'application et appelle le controler adéquat (c'est un front controler). L'autre gère la page de connexion pour le portail client.

8.2 Un objet de gestion d'un élément de liste

Ci-dessous 3 méthodes extraites d'un objet de liste.

On remarque que les méthodes sont très courtes, en effet « Buildings » utilise l'objet pilier « Item » qui agit un peu comme une fonction abstraite.

« Item » implémente de manière générique les méthodes des objets de liste. Tandis que « Buildings » agit comme la spécialisation d'un objet en particulier.

J'ai reproduit ici le mécanisme du polymorphisme offert par les langages orientés objet.

```
Buildings.find = function(itemId) {
  try {
    return Item.findItem(this.STORAGE_KEY, itemId);
  } catch (error) {
    Throw new BuildingException(error.message, "Impossible de trouver
    le bâtiment.");
  }
}

Buildings.updateDependencies = function(itemId, itemDependencies = []) {
  try {
    Item.updateDependencies(this.STORAGE_KEY, itemId,
    itemDependencies);
  } catch (error) {
    Throw new BuildingException(error.message, "Les dependance du
    bâtiment n'on pas pu être mises à jour.");
  }
}

Buildings.addDependence = function(idItem, idDependence) {
  try {
    Item.addDependence(this.STORAGE_KEY, idItem, idDependence);
  } catch (error) {
    Throw new BuildingException(error.message, "L'ajout du lien entre
    le bâtiment et le post a échoué.");
  }
}
```

8.3 La fonction de synchronisation des listes

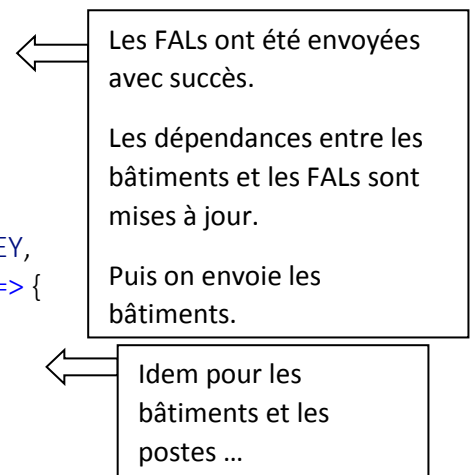
Ci-dessous un zoom sur une partie du code qui permet la synchronisation des éléments de liste. Le code a été volontairement raccourci pour ne garder que la structure globale et simplifier sa lecture.

La fonction est constituée d'un empilement de promesses, qui assurent l'enchaînement des traitements asynchrones dans un ordre bien précis.

En effet, les postes dépendent des bâtiments, qui eux-mêmes dépendent des FALs.

Ainsi, les FALs doivent impérativement être enregistrées en BDD avant les bâtiments, et les bâtiments avant les postes, afin de pouvoir respecter les contraintes de clefs étrangères.

```
publics.syncWithDB = function (callback = null, silent = false) {  
  return new Promise((success, failed) => {  
    // Send all localStorages of dynamics items to DB  
    privates.sendItems(WorkStations.STORAGE_KEY,  
      WorkStations.SERVLET_URL).then((response) => {  
      //Updates all the dependencies  
      //and IDs  
      //Empty updated items array  
      if(App.cancelledSync)  
        failed();  
  
      //Send next itemStorage  
      privates.sendItems(Buildings.STORAGE_KEY,  
        Buildings.SERVLET_URL).then((response) => {  
        //Updates all the dependencies  
        //and IDs  
        //Empty updated items array  
        if(App.cancelledSync)  
          failed();  
  
        privates.sendItems(Posts.STORAGE_KEY,  
          Posts.SERVLET_URL).then((response) => {  
            //Empty updated items array  
  
            On continue en envoyant la liste des activités et  
            des entreprises toute en même temps et on attend pour  
            passer à la suite grâce à « Promise.all ».  
          }).catch((error) => {  
            App.log("ERREUR - L'envoi des données ne s'est  
            pas bien passé. " + (error instanceof Error) ?  
            error.message : "");  
            failed();  
          });  
        });  
      });  
    });  
  });  
}
```



```

        });
    }).catch(() => {
        App.log("ERREUR - L'envoi des données ne s'est pas bien
passé. ");
        failed();
    });
}).catch(() => {
    App.log("ERREUR - L'envoi des données ne s'est pas bien passé.
");
    failed();
});
}).catch(() => {
    App.log("ERREUR - L'envoi des données ne s'est pas bien passé. ");
    failed();
});
});
}

```

8.4 La fonction de la télé-mise à jour.

La fonction « downloadUpdate » a pour rôle de télécharger les fichiers de la dernière version de l'application depuis le serveur. Les fichiers sont téléchargés les uns après les autres.

La synchronisation et le chaînage des téléchargements est possible ici grâce à un mécanisme simple basé sur la récursivité.

En effet la fonction s'appelle elle-même en envoyant en dernier paramètre l'index du fichier qui doit être téléchargé, puis c'est à la ligne surlignée que la condition de sortie de récursivité est effectuée.

```
privates.downloadUpdate = function (filesList, callback, currentIndex = 0) {
  if (!filesList[currentIndex].downloaded) {
    let filePath = filesList[currentIndex].fileUrl.replace(/\\/g, '/');
    let destPath = 'downloads' + filePath.substring(filePath.indexOf('/app/'),
    filePath.lastIndexOf('/'));

    filesList[currentIndex].destPath = destPath;
    DownloadFile.download(App.SERVER_URL + '/../' + filePath, destPath,
    error=> {
      if (!error) {
        filesList[currentIndex].downloaded = true;
        currentIndex++;
        if (currentIndex >= filesList.length) {
          callback(currentIndex, filesList);
        } else {
          callback(currentIndex, filesList);
          privates.downloadUpdate(filesList, callback,
          currentIndex);
        }
      } else {
        filesList[currentIndex].downloaded = false;
        callback(currentIndex, filesList, error);
      }
    });
  } else {
    currentIndex++;
    if (currentIndex >= filesList.length) {
      callback(currentIndex, filesList);
    } else {
      callback(currentIndex, filesList);
      privates.downloadUpdate(filesList, callback, currentIndex);
    }
  }
}
```

Le téléchargement d'un fichier s'appuie sur l'implémentation proposée par une API du moteur Chromium. Il est déclenché par la méthode download de l'objet DownloadFile.

Ci-dessous vous remarquerez l'utilisation de « ipc », qui est l'objet natif d'Electron qui permet de faire communiquer les processus de rendu avec le processus principal. (Voir 3eme schéma du chapitre **8.1** pour plus de précision sur l'architecture d'une application Electron).

L'API qui gère les téléchargements par Chromium fonctionne uniquement dans le processus principal. C'est donc par le billet de « ipc » que la fonction download demande au processus principal de démarrer un téléchargement.

```
"use strict";
```

```
function DownloadException (message, messageIHM) {  
    this.message = message;  
    this.messageIHM = messageIHM;  
}
```

```
Var DownloadFile = DownloadFile || {};
```

```
(function (publics){  
    Let privates = {  
        downloadCallback : null  
    };  
  
    // We listen to download to be completed  
    App.ipc.on("download complete", (event, file) => {  
        App.log("Fichier téléchargé avec succès. (" + file + ')'); // Full file  
path  
        // If any callback was defined, then call it and reset  
downloadCallback  
        if (privates.downloadCallback) {  
            let cb = privates.downloadCallback;  
            privates.downloadCallback = null;  
            cb();  
        }  
    });  
  
    // We listen to the download to failed
```



```

App.ipc.on("download failed", (event, errorMessage) => {
    App.log("[ERREUR] Le téléchargement a échoué.")
    // If any callback was defined, then call it and reset
downloadCallback
    if (privates.downloadCallback) {
        let cb = privates.downloadCallback;
        privates.downloadCallback = null;
        App.log(new Error("[ERREUR] Le téléchargement n'a pas
fonctionné. " + errorMessage).stack);
        cb("Le téléchargement n'a pas fonctionné");
    }
});

/**
 * Starts downloading a file. It throws exception if a file is already
 * being downloaded.
 * The callback receives error message if download failed otherwise
 * no parameter is given.
 *
 * @function download
 * @throws DownloadException
 *
 * @param {string} url The url of the resource to be downloaded
 * @param {string} dest The path where the file should be downloaded
 * @param {function} callback the callback to use when download
 * has finished (failed and success)
 *
 * @returns {void}
 */
publics.download = function (url, dest, callback) {
    if (privates.downloadCallback)
        throw new DownloadException("Already downloading");

    privates.downloadCallback = callback;
    App.log("Début du téléchargement de " + url);
    App.ipc.send("download", {
        url: url,
        properties: {directory: dest}
    });
}
})(DownloadFile));

```

8.5 Pattern MVC

Le pattern MVC est très adapté pour les applications WEB.

Le principe de ce pattern est de découper l'application en 3 types de modules :

- **Modèle** : représente l'état des données à afficher.
- **Vue** : représente de quelle manière les données doivent être affichées.
- **Contrôleur** : contient la logique de l'application ; il fait le lien entre l'interface et les données.

Ce pattern m'a permis de répondre rapidement et efficacement aux modifications et améliorations demandées par le client tout au long de la réalisation de l'application.

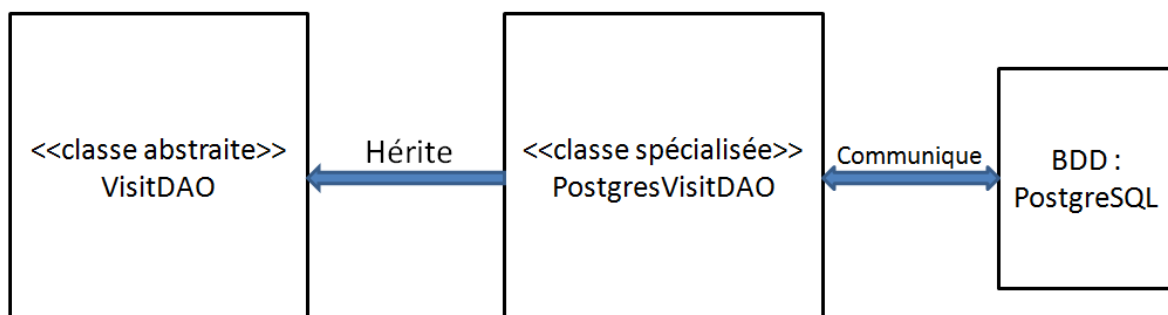
Il est vrai que bien organiser et bien ranger son projet prend beaucoup de temps. Mais une fois en place, on gagne un temps considérable lors du maintien et de l'amélioration du projet.

8.6 Pattern DAO

La communication entre la BDD et l'instance J2EE est assurée par des Data Access Object.

Pour garantir une évolution peu contraignante, les DAOs héritent et implémentent des classes abstraites qui font office d'interface.

Exemple pour le DAO Visit :



8.9 Pattern singleton

Les DAOs sont thread-safe. En effet tous les attributs sont déclarés « final ».

Ainsi la DAOFactory fournit une instance unique de chaque DAO. Cela évite la surcharge inutile d'instance de classe en mémoire.

8.10 Module pattern

Afin d'éviter les conflits d'accès aux ressources et de profiter de la puissance qu'offre la programmation orientée objet, j'ai opté pour le module pattern.

Le module pattern offre la possibilité de reproduire le mécanisme d'encapsulation par le billet des portées de variables.

Le module pattern repose sur une fonction auto-invoquée qui retourne une propriété qui contient tout ce qui doit être publique.

Pour ma part j'ai utilisé une méthode un peu différente pour récupérer une référence sur les attributs et méthodes publiques.

Je l'ai fait en passant une variable globale en paramètre de la fonction.

```
var module = module || {};  
  
(function (publics) {  
    let private = {};  
  
    private.methodePrivee = function () {...}  
  
    public.methodePublique = function () {...}  
})(module);
```

9. Gestion de projet

9.1 Définition des objectifs et jalons

Etant donné que mon stage a été coupé en 2 parties (une en Juillet puis l'autre en novembre et décembre), il a été convenu d'un premier jalon : une version 1 qui doit être livrée à la fin de la première période.

Cette version doit uniquement proposer l'application mobile fonctionnant en local et permettant d'enregistrer des visites, puis de les ressortir au format CSV exploitable avec Excel.

Fin décembre, il est prévu que la version 2 soit livrée et puisse prendre en charge la totalité du besoin client exprimé.

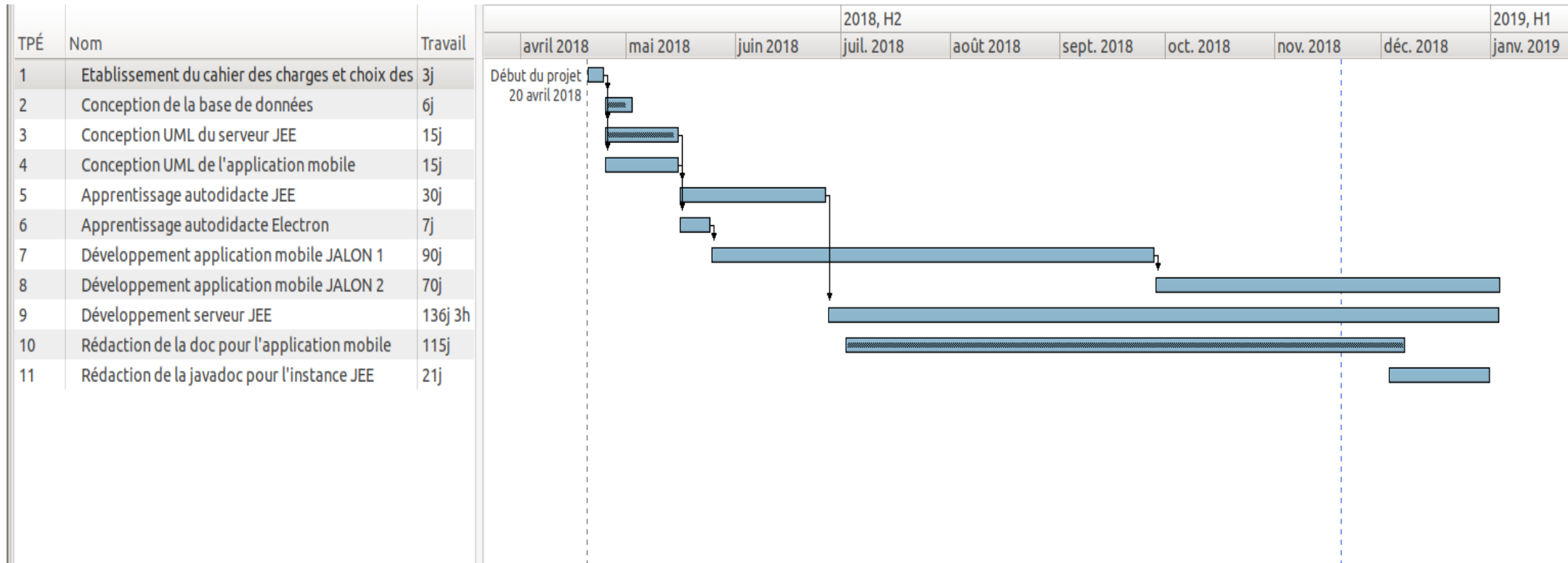
9.2 Phases de test

Durant le stage, une réunion hebdomadaire a été réalisée avec mon tuteur afin d'étudier l'avancement du projet et de lever les points bloquants.

Durant tout le mois de décembre 2018, l'Ingénieur Sécurité a utilisé l'application sur le terrain et m'a remonté les bugs et suggéré les améliorations nécessaires.

9.3 Planning

Ci-dessous un diagramme de Gantt regroupant les principales étapes de conception et de développement de l'application.



10. Retour d'expérience

Le fait que l'application doit pouvoir fonctionner en mode hors ligne contraint à dupliquer la donnée, tout en gardant une structure et une organisation robuste et bien pensée.

La gestion de l'envoi des photos et leur affichage sont des fonctionnalités qui ont demandé une semaine et demi de travail. Cela n'a pas été simple pour 2 raisons :

- Sur l'aspect technique : il a fallu apprendre à utiliser les flux.
- Sur le plan organisationnel : il a fallu jongler entre les chemins d'accès relatifs et absolus, puis entre la version mobile et WEB de l'application.

Ensuite, l'interface de « consultation » m'a donné beaucoup de fil à retordre. En effet, il a fallu être capable d'afficher un maximum d'informations à l'écran tout en restant clair, ergonomique et intuitif.

11. Conclusion

Je suis fier de l'application que j'ai réalisée. J'ai fait plus que je n'avais espéré et que le besoin client exprimé, en réalisant un module capable de mettre à jour l'application automatiquement, et en peaufinant les IHM pour les rendre les plus ergonomiques possible.

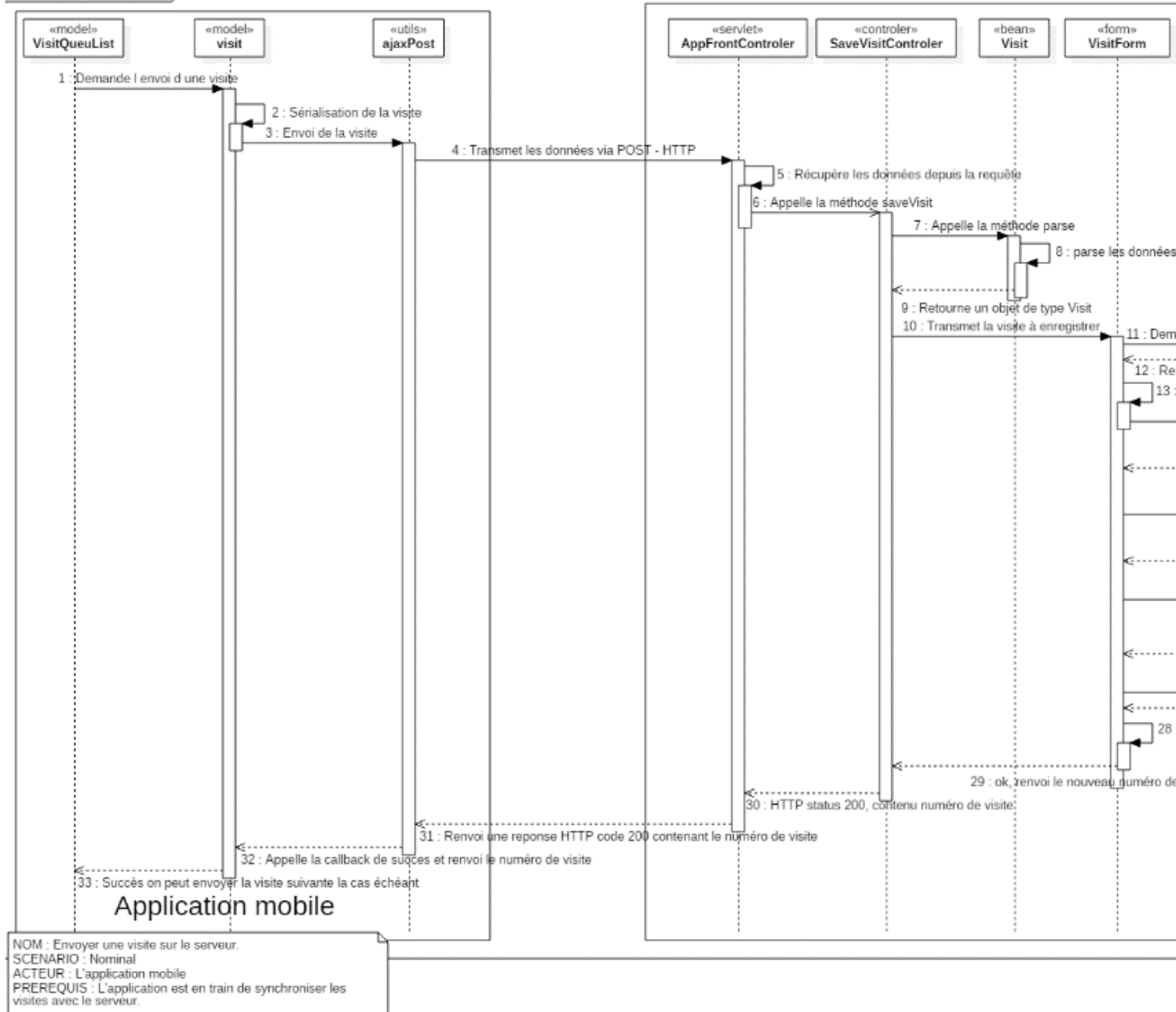
Ma plus grande satisfaction fut quand l'opérateur de chez Trigo Qualitaire m'a félicité pour mon travail, et lorsque j'ai appris qu'Airbus trouvait l'application si fonctionnelle qu'il a demandé à Trigo de transférer toutes les données depuis le début de la prestation (2016) sur l'application.

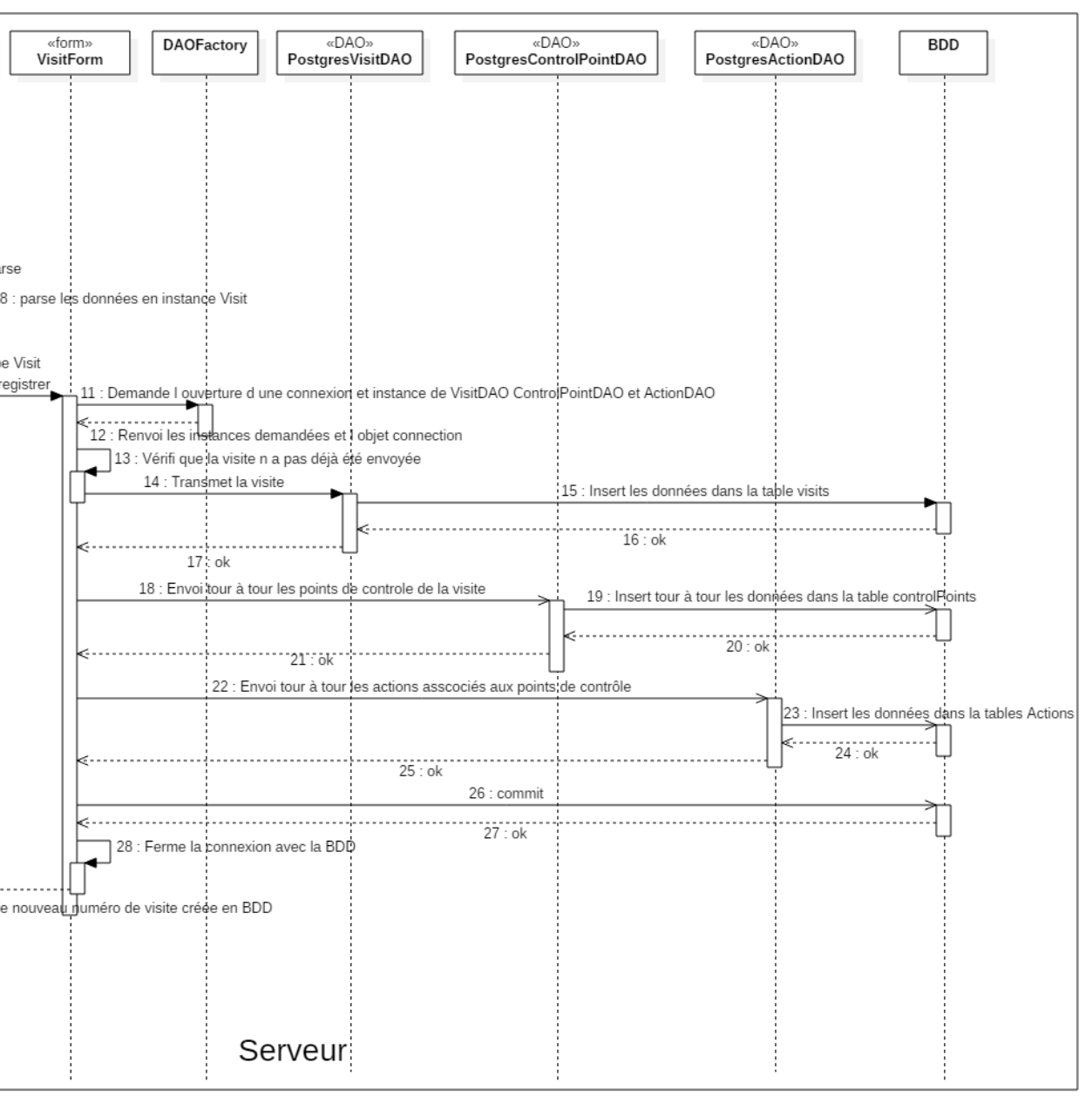
Cette application est pour moi une belle carte de visite à ajouter à mon CV afin de concrétiser ma reconversion professionnelle.

12. ANNEXES

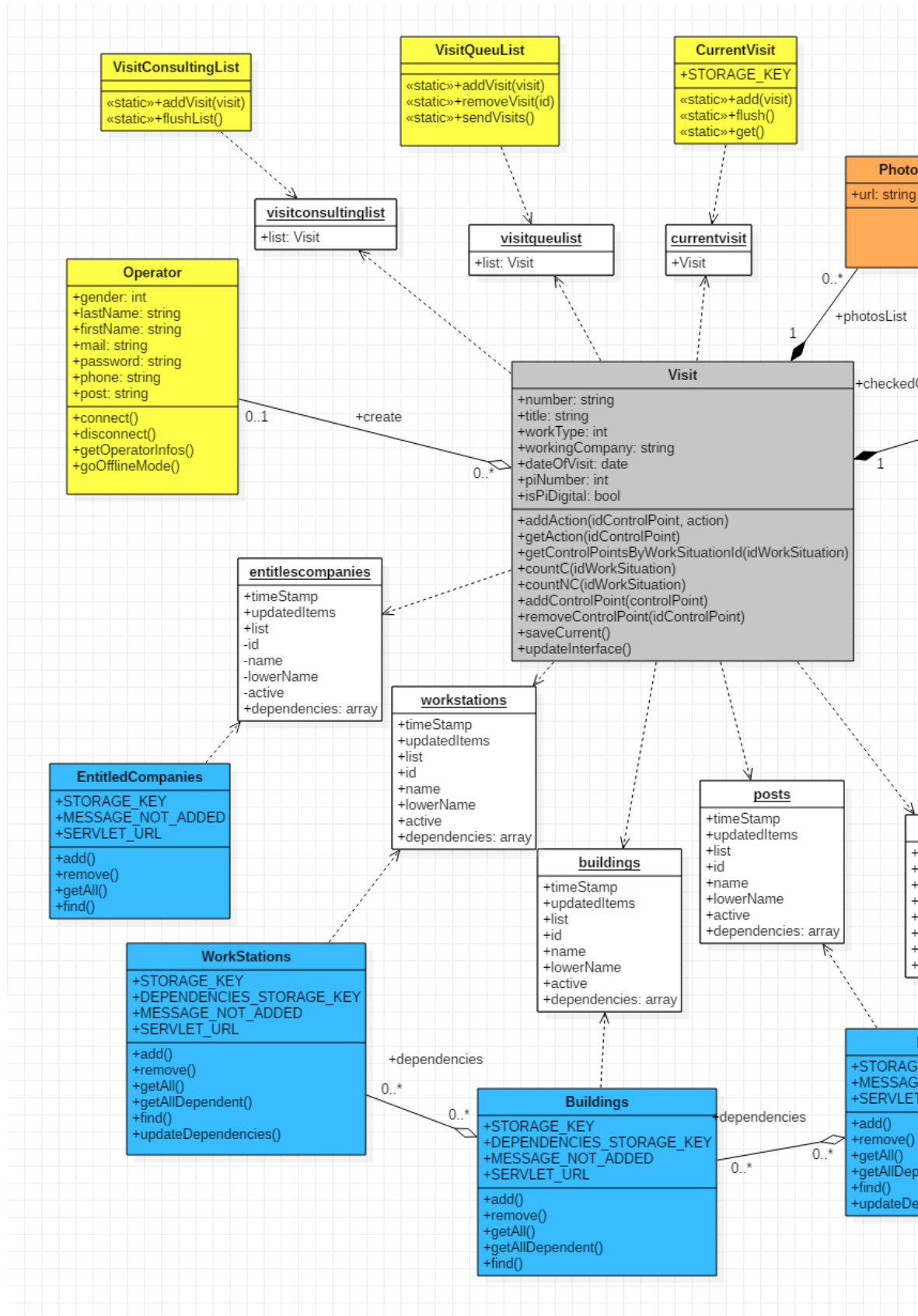
12.1 Annexe 1

Interaction Envoi d'une visite

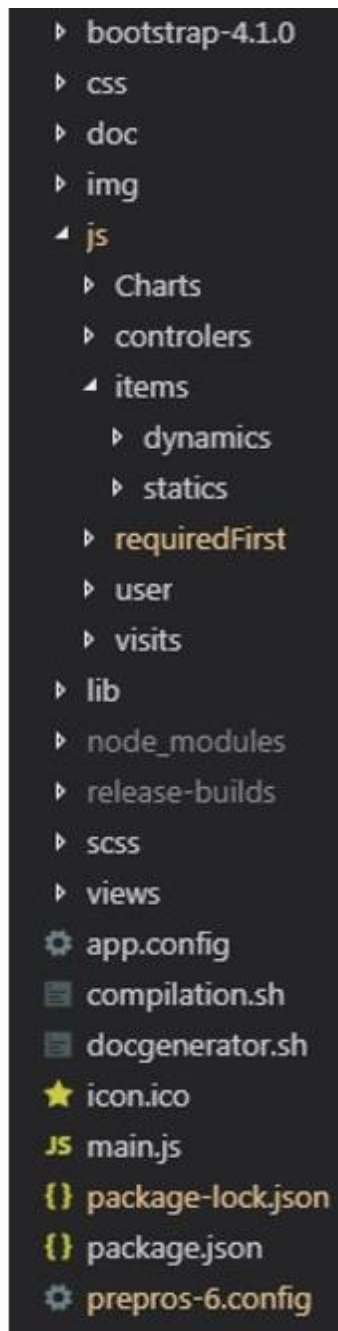




12.2 Annexe 2



12.3 Annexe 3



- **bootstrap** : Version développement du Framework Bootstrap. Me permet ainsi de recompiler bootstrap.css avec uniquement les besoins minimums pour le projet.
- **css** : Regroupe les feuilles de style (une par vue).
- **doc** : Regroupe la documentation des classes de l'application.
- **img** : Toutes les images et logos utilisés dans les diverses interfaces de l'application.
- **js** : Regroupe par catégories tous les scripts JavaScript.
- **lib** : Toutes les librairies utilisées par l'application.
- **node_modules** : Modules de node gérés par le Node Package Manager.
- **release-builds** : Dossier de destination des versions compilées de l'application.
- **scss** : Regroupe les versions non compilées (Sass) des feuilles de style.
- **views** : Regroupe toutes les vues (HTML).
- **app.config** : Permet le paramétrage technique de l'application.
- **compilation.sh** : Script bash permettant d'automatiser le processus de compilation.
- **docgenerator.sh** : Script bash permettant de lancer la génération de la documentation du code source.
- **main.js** : Point d'entrée de l'application (Voir architecture d'une application ElectronApp)
- **package.json & package-lock.json** : Manifeste pour le Node Package Manager.
- **prepros-6.config** : Fichier généré par Prepros, le gestionnaire de compilation pour Sass et JS.

