Tempora 3 - Message général

Cahier des charges

Stage du 02/12/2024 au 24/06/2025

Introduction

Le présent cahier des charges est à destination d'une équipe de stagiaires et concerne l'amélioration d'une fonctionnalité sur la version 3 de l'application Tempora, développée par la société Tempora SAS. Ce document contiendra des informations au sujet du projet, de l'existant et de l'entreprise.

Il débutera par le contexte du projet et par ses objectifs. Dans un deuxième temps, il présentera Tempora dans sa globalité (l'entreprise, l'équipe et l'application). Afin de bien comprendre le projet, il donnera une brève présentation de ce qui avait été réalisé. Pour finir il donnera les informations relatives au projet le plus détaillé possible tout en laissant une marge de manœuvre au développeur.

Contexte

La société Tempora est un éditeur d'application qui propose à plusieurs milliers de professionnels du domaine médical, paramédical, libérale et autre, une solution applicative d'agenda partagé. Cette application existe depuis 1998 et la version actuelle de l'api est utilisée depuis 2016. Pour la période d'août 2024, Tempora dénombre plus de 1000 agendas actifs.

L'application de Tempora met à la disposition de ses utilisateurs une multitude de fonctionnalités qui vont de la prise de rendez-vous et de message à destination des professionnels, à l'envoi de consignes aux secrétaires.

L'équipe de Tempora reçoit régulièrement des demandes de mise à jour de la part de ses clients, l'une d'entre elles est notamment la possibilité d'envoyer un message général à une liste de clients prédéfinis. L'application permet déjà aux permanences téléphoniques d'envoyer un message à tous leurs clients. Elle permet aussi de regrouper les dossiers par groupes d'organisation. Le projet aura donc pour but final de permettre l'envoi de messages sur tous les dossiers appartenant à des groupes qui auront été sélectionnés.

Objectifs

Le projet aura plusieurs objectifs dont certains seront secondaires. Les objectifs principaux sont :

- Envoyer un message à une liste de groupes sélectionnés.
- Envoyer un message à une liste d'organisations (sans groupe) sélectionnées.
- Améliorer la page d'envoi du message général, l'interface est un peu obsolète.

Les objectifs secondaires sont :

- Améliorer l'envoi de messages généraux, pouvoir personnaliser les informations recueillies.
- Utiliser les messages type de la permanence.
- Réaliser une documentation technique pour la mise en ligne des modifications.
- Proposer d'autres améliorations du module.

Qu'est ce que Tempora ?

L'application

L'application Tempora est une application d'agenda partagé entre le secrétariat et les professionnels propriétaires d'un agenda. Cette application en est actuellement à sa troisième version.

Tempora 1, a été développée en 1998 par Mr Joël Connault. Ce développement à eu lieu pour répondre au besoin de la société Allo Supletel, une permanence téléphonique à Brest, aujourd'hui toujours cliente de la société Tempora. Cette version de l'application était à la fois disponible en tant qu'application native et web. Pour la version native le langage utilisé était le Delphi. Deux SGBD[1] étaient utilisés : Paradox pour la version locale et MySQL pour la version web. Cependant, Tempora 1 était très contraignante car elle demandait l'installation d'un serveur chez les clients ainsi qu'une synchronisation des données faite manuellement par l'utilisateur. La mise à jour de l'application native pouvait poser certains problèmes car elle devait être réalisée individuellement sur chaque poste utilisateur.

Tempora 2, a été réalisé en 2010. L'avantage de cette version est qu'elle demandait l'utilisation d'un serveur unique, réglant ainsi le souci de mise à jour. Elle demandait l'utilisation d'un applet Java sur chaque poste où l'application serait utilisée, ce qui nécessitait l'utilisation d'un navigateur web. Contrairement à la version précédente, cette version n'utilisait plus qu'un seul SGBD. Elle a aussi apporté une nouveauté par rapport à Tempora 1, la synchronisation des données en temps réel.

C'est suite à un problème de support des applets Java par les navigateurs que Tempora 3 a été réalisé. Cette version a été mise à disposition des clients en 2016. Elle a nécessité la refonte complète de l'application, avec changement de structure, de langage et de SGBD. Le développement initial a été réalisé par une société de sous-traitance.

[1] Serveur de Gestion de Base de Données

L'application étant une application web avec utilisation d'une API REST, les langages utilisés sont les suivant :

- HTML 5, CSS et JavaScript pour l'interface utilisateur
- Scala pour la partie api.

Le SGBD choisi est Postgres SQL. Tempora 3, est la version actuellement proposée aux clients.

En 2019, une version dérivée de Tempora 3 a été mise à disposition des clients. C'est une version simplifiée et mobile. Elle à été réalisée en utilisant la technologie hybride en passant par ionic. C'est-à-dire qu'elle a été développée comme une application web en utilisant HTML 5 et compiler comme une application native disponible sur les téléphone et tablette Android et IOS.

Depuis mai 2023, la société Tempora a débuté le développement de Tempora 4.

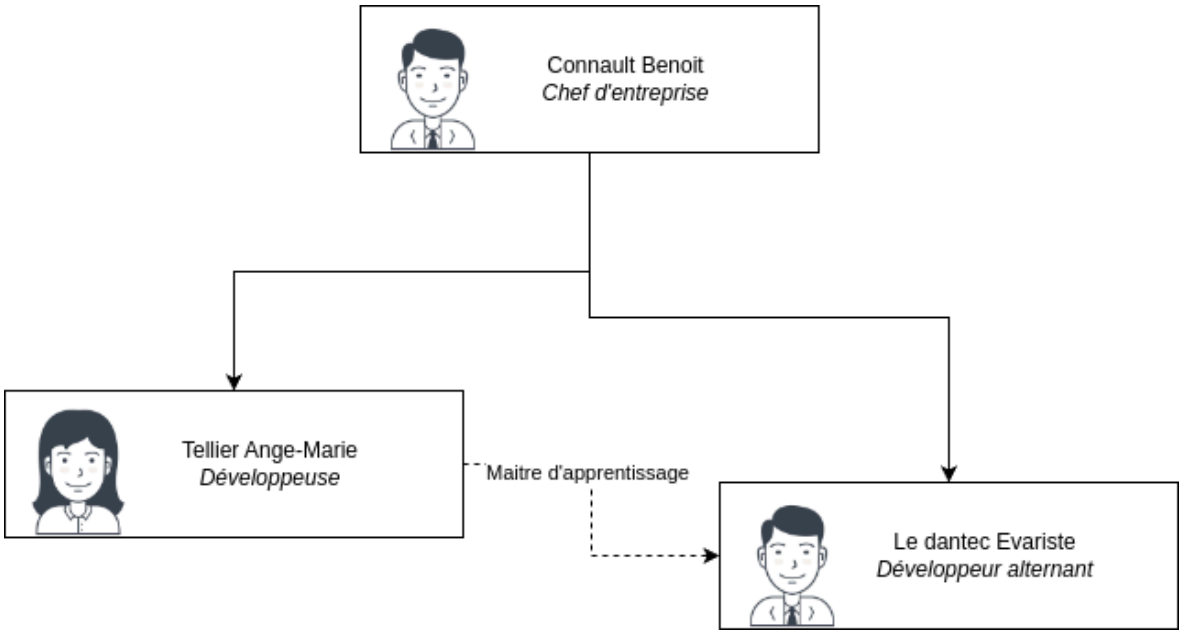
L'entreprise et son équipe

La société Tempora à été fondée en 2009 afin de pouvoir proposer Tempora à des clients externes à Allo Supletel. L'équipe de Tempora est actuellement composée de 2 personne à temps plein :

- Connault Benoit en sa qualité de chef d'entreprise et de développeur. Pour son parcours scolaire, il a réalisé un Bac STI, et une première année en BTS informatique, qui a malheureusement été écourté suite à un problème financier de l'école. Après cela, il a suivi quelques périodes de formation non diplômante. Il a commencé un contrat professionnel avec Allo Supletel quelques années avant la création de la société Tempora. Il a commencé par faire de la maintenance et du dépannage sur les deux versions de l'application. Il a aussi réalisé quelques opérations d'amélioration de Tempora 2. Lorsque le sous-traitant qui devait réaliser le développement initial a pris du retard, Benoit a participé à ce dernier. Depuis, il réalise des améliorations continues. Entre septembre et octobre 2020, il a racheté l'entreprise de Tempora après le départ du précédent chef d'entreprise.

- Tellier Ange-Marie en sa qualité de développeuse. Son parcours scolaire est assez atypique, elle a réalisé un Bac Pro Service à la personnes dans un lycée agricole. Puis elle a poursuivie ses études dans l'informatique en réalisant un BTS SIO et une formations de deux ans en Bac +4, avec le CESI de Brest. C'est pendant cette formation qu'elle a signé un contrat de professionnalisation en alternance avec Tempora, en 2016. Et a continué son travail au sein de l'entreprise avec un CDI depuis 2018.

Depuis septembre 2023, une nouvelle personne, Le Dantec Evariste, a rejoint l'équipe de Tempora en tant que développeur en alternance.



Organigramme de Tempora depuis septembre 2023

Une synthèse des dates concernant l'application et l'entreprise est disponible en Annexe 1.

Étude de l'existant

L'application Tempora se divise en deux projets globaux : le frontend et l'api. Comme mentionné précédemment, la version actuellement en production à été mise à disposition des clients en 2016.

L'application a été conçue pour répondre aux besoins de divers professionnels, que ce soit des professionnels de santé, des avocats, des notaires ou autres. Elle est, de manière générale, à destination des permanences téléphoniques. Elle est composée de 4 modules principaux : la prise de rendez-vous (l'agenda), la messagerie, les consignes et le répertoire. Les messages sont liés aux agenda, et il peut arriver que les permanences ai besoin d'envoyer le même message à la totalité de leur clients, c'est ici qu'intervient la fonction d'envoi de message général. Mais avant d'entrer dans le vif du sujet, voici une description et une étude de l'existant.

L'api

L'api de Tempora est une api REST, c'est-à-dire, de manière simplifiée, qu'elle peut être utilisée par d'autres applications, s'ils ont les routes d'accès et des droits pour l'utilisation. Actuellement toutes les applications de Tempora (le frontend, la prise de rendez-vous en ligne, les applications mobiles) en production utilisent cette API.

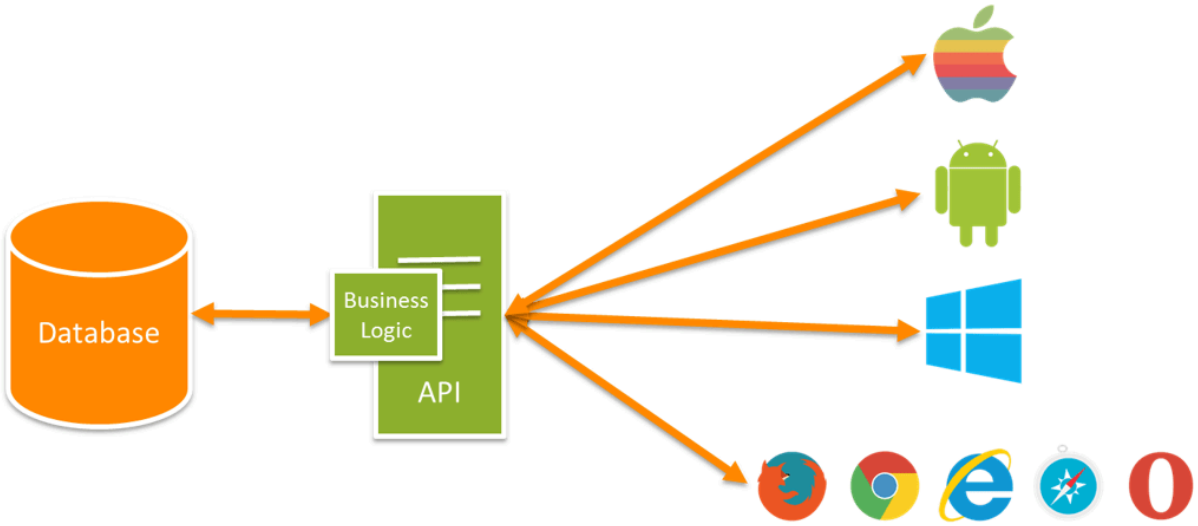


Schéma d'une API REST

Or depuis quelques temps, tempora-backend (l'API actuel) rencontre quelques soucis pour l'évolution de l'application. C'est pour cela que le projet de l'API v4 a été mis en place. Les développements nécessaires à la réalisation du projet seront donc réalisés par Ange-Marie ou Benoit. Et devront être aussi reportés sur la v4.

Les technologies utilisées

L'API, comme mentionné précédemment, a été développée en scala. Le scala est un langage à la fois objet et fonctionnel, dérivé du Java. La version de scala actuellement utilisée est la 2.11.1. Pour ce projet, les développeurs utilisent le framework play qui est sur la version 2.3.10. Ces deux versions sont obsolètes et seront mises à jour avec la version 4 de l'api, qui est en cours de développement. L'api 3 utilise aussi une version ancienne de java, la 8.

L'architecture du projet

L'architecture globale de tempora-backend est faites en MVC[2] (voir dans la partie concernant le projet) avec quelques petites subtilités qui seront expliqués ensuite.

[2] Model View Controller

Le projet possède plusieurs dossiers tels que app, conf, lib, etc... Le dossier conf, comme son nom l'indique, contient la configuration du projet et notamment les fichiers de routes qui permettent de rediriger les requêtes faites par le frontend. Le dossier app contient la totalité des fichiers nécessaires au fonctionnement du projet, c'est-à-dire les contrôleurs, les modèles (dbdata), les fichiers d'api, et les vues.

- **Les fichiers d'api**
  Ils servent d'interface entre Tempora et d'autres applications externes comme google calendar, par exemple.

- **Modèles**
  Ils font l'interface entre la base de données et les contrôleurs. Ils exécutent les requêtes SQL en fonction des demandes faites par les contrôleurs. Puis ils traduisent le résultat obtenu en objet à fournir à ce dernier.
  Ce sont eux qui contiennent la structure d'une classe objet et les fonctions de conversion pour ces dernières.

- **Contrôleurs**
  Ce sont les fichiers au centre des requêtes envoyées par le frontend. Il les reçoivent par le biais des fichiers de routes et font appel au dbdata pour les exécuter dans la base de données.
  Ils traitent ensuite le résultat obtenu et renvoient une réponse au frontend.

- **Vues**
  Ce sont des fichiers HTML, permettant d'envoyer des informations par mails à la suite d' une requête de l'api. Ou bien d'avoir un aperçu avant l'impression.

Pour en savoir plus sur la communication entre les différents fichiers du projet voir l'Annexe 2.

Le frontend

Les technologies utilisées

Le projet tempora-frontend, est un projet web classique qui utilise :

- HTML pour l'IHM[3].
- CSS, en passant par du less pour les feuilles de style. L'un des avantages de less, utilisés dans le projet, est qu'il permet l'imbrication de blocs. Par exemple, il est possible de mettre un style général à une div et d'ajouter un bloc pour dire que si

[3] Interface Homme Machine

l'une d'entre elle à la classe "A", un ou plusieurs éléments de son style doivent changer.

- Javascript en utilisant jQuery pour le côté fonctionnel de l'application. Le projet utilise aussi de nombreuses librairies et utilitaires toutes basées sur jQuery qui permettent de faire des raccourcis et ainsi éviter de faire de la répétition de code.

La communication avec le backend

Tempora étant une application dynamique avec une API REST, elle doit envoyer des requêtes à cette dernière afin d'envoyer ou de récupérer des données. Et ce parfois sans avoir à recharger la page. Pour cela, les développeurs utilisent Ajax.

Ajax n'est ni une technologie ni un langage de programmation, il s'agit d'un concept qui reposent sur plusieurs technologies tel que json, XML et javascript. Comme mentionné précédemment, il consiste à envoyer des requêtes au serveur sans avoir besoin d'exécuter un rechargement de la page. C'est d'ailleurs pour cela que la plupart des développeurs vont préférer utiliser javascript pour réaliser des requêtes ajax.

Pour simplifier le codage des requêtes ajax, Tempora 3 utilise jQuery et a mis en place une fonctionnalités qui demandent des paramètres tels que la route et le type de requête entre autres choses.

Pour en savoir plus sur Ajax https://www.brioude-internet.fr/8454/qu-est-ce-que-l-ajax/

L'architecture du projet

Comme pour l'API, le frontend utilise globalement une architecture MVC, avec quelques particularités et fichiers supplémentaires. Le développement sur ce projet concerne principalement trois types de fichiers qui sont : les contrôleurs, les managers et les vues. Voici leurs descriptions :

- **Contrôleurs**
  Ils sont le cœur du logiciel et permettent de gérer les aspect de Tempora.
  C'est par le biais des contrôleurs que les demandes de données sont faites. C'est aussi eux qui les traitent et les fournissent au vue pour être affichée. Ce sont les contrôleur qui déclenche l'affichage d'un fichier HTML lorsque cela est nécessaire.
  Ils reçoivent et traitent les demandes utilisateurs au travers d'événements fournis par un autre fichier de l'application.
  Un contrôleur "connaît" et peut appeler n'importe quel manager et vue js. Généralement un contrôleur est dédié à un seul fichier HTML. Il peut arriver que

dans certains cas, comme les formulaires de création et d'édition d'un élément, un fichier HTML soit relié à deux contrôleur, un pour chacune des deux action.

- **Managers**
  Ils font la passerelle entre les contrôleurs et l'API. manager est instancié et appelé par ce premier. Le contrôleur souscrit à un événement déclenché par le manager pour attendre la réponse à sa requête.
  Ce dernier transmet la demande au serveur par le biais d'une requête Ajax. Quand il reçoit la réponse, il déclenche l'événement de succès ou d'erreur auxquels est inscrit le contrôleur afin qu'il puisse traiter la réponse.
  L'adresse du serveur sur lequel les requêtes API sont exécutées est enregistrée dans un fichier afin de ne pas avoir à parcourir toutes les fonctions pour la changer si besoin est. Le fichier en question se nomme api.js. Pour les tests l'adresse du serveur est bien entendu localhost, pour cela api.js fournie deux variables : "serveur" pour la production et "local" pour les test et le dev.
  Un manager ne peut faire référence qu'à lui-même, il ne connaît pas les contrôleurs, vues etc.. qui font appel à lui.

- **Vues**
  Une vue dans Tempora 3, correspond à un fichier js qui gère l'affichage dynamique d'une page. Elle reçoit les données de la part du contrôleur et les affiche en modifiant le contenu du html.
  Elle met également en place des événement pour réagir aux commandes de l'utilisateur, afin qu'elles soient traitées et, si besoin, transmises au contrôleur en générant un autre événement, qui sera déclenché dans un fichier secondaire et pourra transmettre la demande.
  Généralement une vue est spécifique à un seul contrôleur et donc un seul html, mais il peut arriver que dans certains cas, une vue correspondent à plusieurs contrôleurs, comme mentionné dans la description des contrôleurs. Par défaut, une vue ne connaît qu'elle-même et les fichiers html qui auront son sélecteur.

Pour comprendre vraiment la correspondance entre les différents fichiers du frontend et la communication avec l'api, voir l'Annexe 3.

Le projet

Il a précédemment été établi que la fonction d'envoi de messages générale ne permettait pas de choisir les organisations ou groupes auxquels le message doit être envoyé. C'est dans le but de rajouter cette possibilité que ce projet à été mis en place. L'équipe de développement chargé de ce projet travaillera uniquement sur la partie frontend.

L’équipe projet

Comme pour tout projet de développement, celui-ci à besoin d’une équipe de développeurs. Chaque membre de l’équipe aura un rôle prédéfinie, cette répartition n’est pas exhaustive et pourront évoluer. Les rôles sont les suivants :

- Chef de projet
- Développeur/Développeuse principale
- Développeur/Développeuse (stagiaire ou non)
- Alpha testeur

Même si dans ce projet, certains membres de l’équipe se verront attribuer le rôle d’alpha testeur, cela n’enlève en rien le fait que le développeur ou le développeur principal teste son code avant de le soumettre.

Les membre de l’équipe de développement et leur rôles sont les suivant :

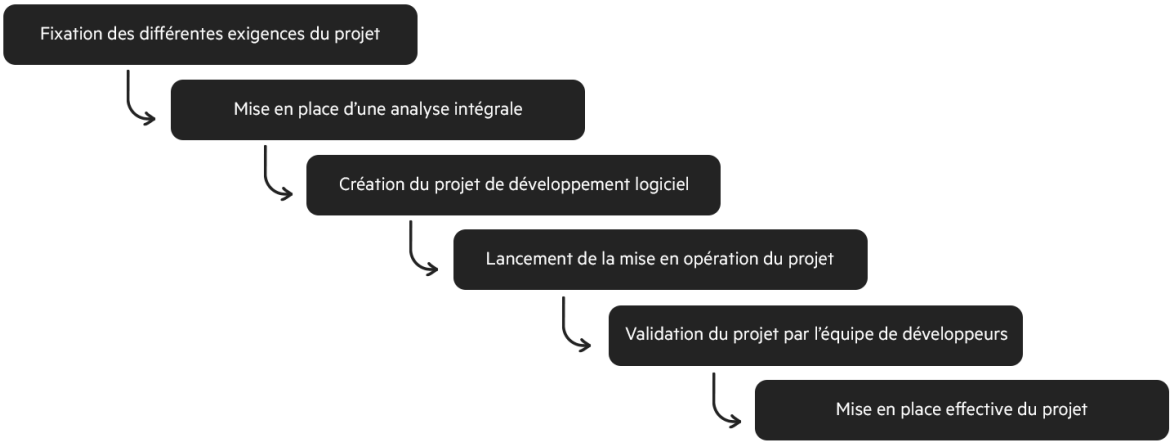
- Tellier Ange-Marie : Chef de projet, développeuse principale et alpha testeur.
- Connault Benoit : Développeur principal et alpha testeur.
- Bialgues Marion : Développeuse
- Médoc Adèle : Développeuse

La méthode de développement de l’équipe

Il existe plusieurs méthodes de développement, les plus connues sont : la méthode en cascade, la méthode lean, la méthode itérative et la méthode agile (Cf. https://www.eurotechconseil.com/blog/quelles-sont-les-methodologies-de-developpement-logiciels/)

La méthode en cascade

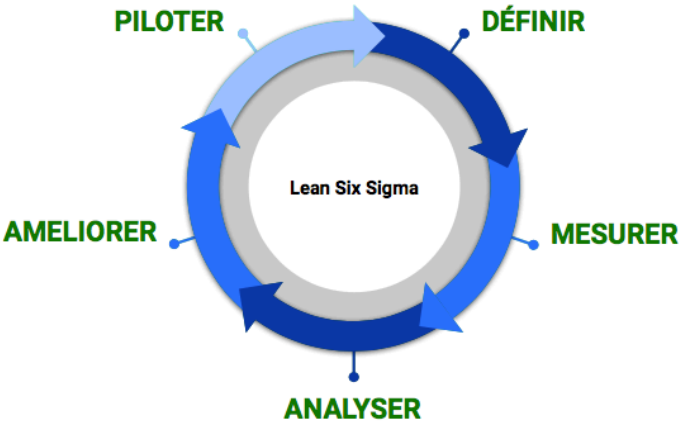
C’est une méthode fragmentée en 6 étapes. Elle a pour avantage d’avoir un passage rapide entre les différentes étapes et d’avoir un processus pertinent. Cependant, cette méthode n’est pas adaptée à la manière de travailler qu’utilise la société Tempora, car elle ne permet pas de revenir sur une étape.



La méthode lean

Cette méthode s'inspire "des procédés et activités de conception éthique". Elle repose ces fondamentaux :

1. la suppression du gaspillage
2. le renforcement de l'apprentissage
3. le report de la prise de décision
4. la livraison rapide
5. la responsabilisation des développeurs
6. l'intégration du processus

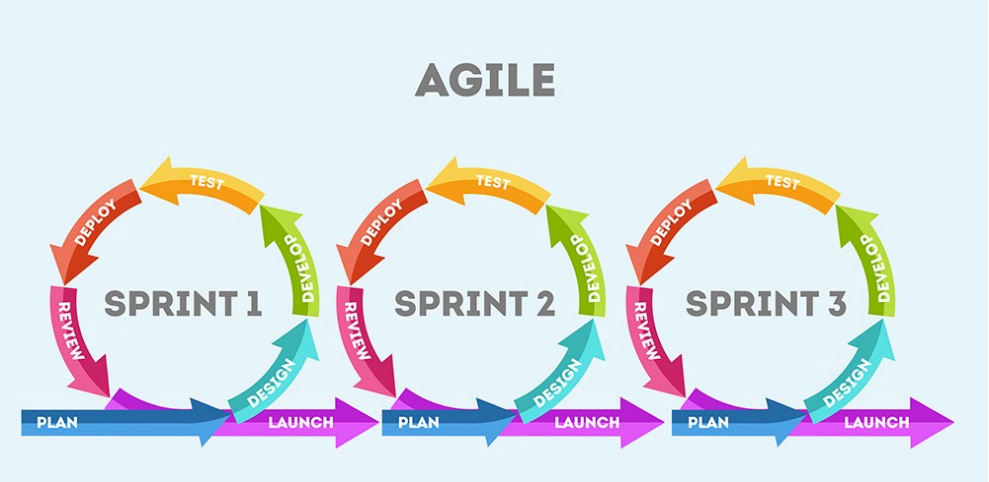


Cette méthode de développement ne donne pas lieu au multi-tâche, car elle consiste à se concentrer sur une tâche à réaliser à un moment T. Avec cette méthode, il est question "d'attendre la performance en matière de qualité de produit, de délais de livraison, de productivité ainsi que de réduction des coûts. Des objectifs qui peuvent être atteints grâce à un process d'amélioration soutenue ainsi que la suppression du gaspillage".

Le simple fait de ne pas pouvoir faire du multi-tâche retire l'intérêt de cette méthode pour l'équipe de Tempora, qui pourrait avoir besoin de travailler sur différentes parties du projet de manière simultanée.

La méthode agile

La méthode de développement agile permet une gestion de projet itérative et collaborative. Elle met l'accent sur la flexibilité, la réactivité et l'adaptation au changement au sein même de l'équipe et du projet. Elle favorise une collaboration étroite entre tous les membres de l'équipe et encourage la communication ouverte et constante. Ce qui permet de s'assurer que les attentes sont clairement définies et comprises.

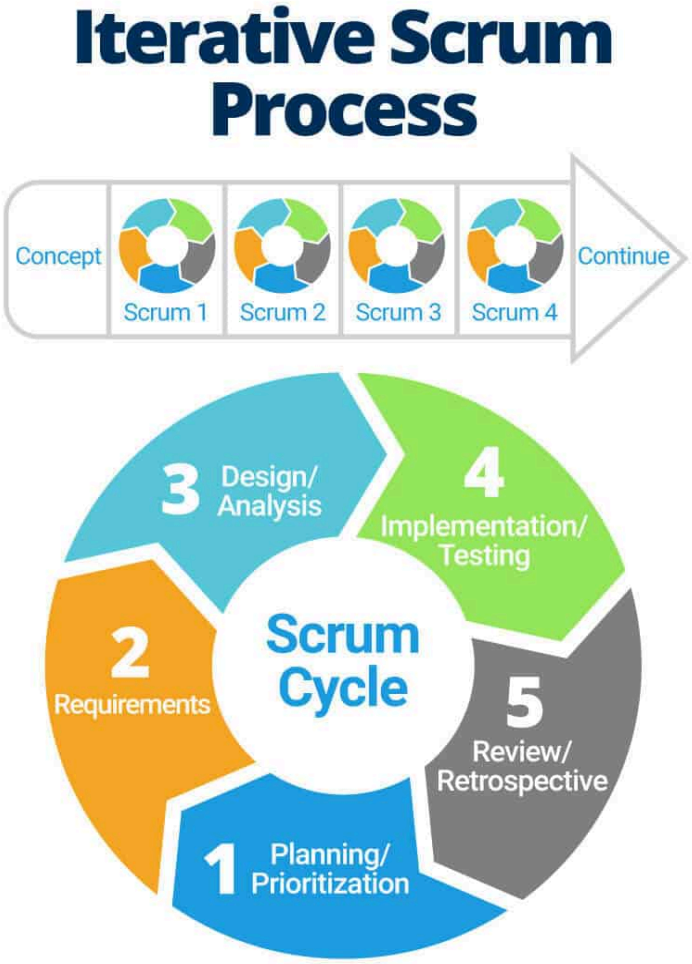


Cette méthode est adaptée à l'équipe de Tempora, car elle permet de se concentrer sur une fonctionnalité à livrer tout en permettant l'adaptation et les changements fréquents. Elle permet aussi de revenir en arrière sur une itération tout au long du processus de développement, ce qui permet une adaptation du code et des fonctionnalités en cours de développement.

Les informations sur cette méthodologie, ont été trouvé sur le lien suivant : https://www.eurotechconseil.com/glossaire/agile/

La méthode itérative

Cette méthode ressemble beaucoup à l'agile, qui utilise aussi des itérations. La différence réside dans le fait que la méthode itérative satisfait deux besoins essentiels : le respect des délais de livraison et la détermination des coûts du projet à l'avance. Elle se concentre également sur la possibilité d'avoir accès à une frange d'innovation.



Cette méthode de développement s'adressera plutôt aux équipes qui ont tendance à préférer les méthodes d'organisation classique et qui interviennent dans un contexte plutôt rigide. Cette méthode s'adresse principalement aux projets qui ont un cahier des charges défini avec peu de flexibilité.

Les points relevés dans le paragraphe précédent, à savoir le manque de flexibilité dans la gestion du projet et du cahier des charges, font que cette méthode n'est pas adaptée à l'équipe de Tempora et au projet actuel.

En conclusion, l'équipe de Tempora travaillera plutôt avec la méthode agile.

L'environnement de travail

Pour développer l'environnement de travail des membres de l'équipe est principalement constitué d'un poste de travail sous Linux ainsi que de nombreux outils utiles à la gestion de projet, le développement et de communication. Cependant, si le poste physique est sous Windows ou Mac, l'entreprise peut fournir une machine

virtuelle Ubuntu (système d'exploitation linux) et permettre au développeur de travailler en ssh[4].

Les outils utilisés par l'équipe et le projet

Certains des outils mentionnés ci après, ne sont pas obligatoires : les membres de l'équipe peuvent en choisir d'autres. Seuls les outils de gestion de projet et de communication sont non négociables.

Pour développer, l'équipe de Tempora utilise généralement vsCode qui peut être télécharger en suivant le lien suivant : https://code.visualstudio.com/download

Pour ce qui concerne la communication distante, elle se fera principalement par discord, depuis le serveur de la société sur les salons de la catégorie "Stages". L'échange de fichier quant à lui sera fait par le drive si ce n'est pas un fichier code. Pour ces derniers nous utiliserons l'outil de gestion de projet et de versionning git.

Pour gérer au mieux les tâches à accomplir, leur répartition et leur durée, l'équipe de Tempora utilise l'application web yookkan, disponible gratuitement en suivant le lien suivant : https://app.yookkan.com/. Dessus se trouve un tableau sur lequel les membres de l'équipe de développement sont ajoutés. Le tableau est composé de 5 colonnes :

- A faire : Pour les tâches à réaliser, qui ne sont pas encore attribuée.
- En cours : Il ne faut pas hésiter à mettre une date de début et une date de fin en fonction des capacités du développeur. Il faut aussi modifier le niveau d'avancement de chaque tâche en cours afin de permettre au chef de projet et aux autres membres de l'équipe d'avoir une idée de l'avancement de vos tâches et du projet en sa globalité. Attention : La durée d'une tâche doit être établie en fonction des capacités réelles des membres de l'équipe qui en sont chargés. De plus, si une tâche est dépendante d'autre tâche en amont ou que des cartes (tâches) en aval de celle- ci dépendent d'une fonction ou d'un morceau de code, il faut que ce dernier soit réalisé en priorité, cela évitera de prendre du retard.
- Réalisé : Pour les tâches terminées. Attention une tâche terminée n'est pas forcément validée et peut revenir dans la case WIP.
- Validé : Pour les tâches qui sont finies et validées par le chef de projet et les alphas testeurs.

Pour la modélisation des UML, avant le développement Modelio sera à utiliser.

[4] Secure Shell, protocole sécurisé de communication entre machine (virtuelle et/ou physique). Cf Annexe 4 pour utiliser ssh

Installation de l'environnement de dev

Avant d'installer l'environnement de développement, il est demandé d'avoir installer git et de l'avoir paramétrer. L'installation et le paramétrage de git doivent se faire depuis la machine linuxe que ce soit une machine physique ou virtuelle. Pour se connecter à la VM (Machine virtuelle) en ssh voir l'annexe 4.

Installer git

Vérifier la version de git installé

```
$ git --version
```

Installer git sur le poste linuxe

```
$ sudo apt-get install git
```

Configuration locale de git

```
$ git config --global user.name "John Doe"
```

```
$ git config --global user.email johndoe@example.com
```

Attention : Remplacer John Doe par votre nom et johndoe@example.com par votre email git.

Configuration de gitlab

Pour configurer votre compte gitlab avec une clé ssh, se connecter sur gitlab et suivez la procédure suivante :

1. Aller sur votre profil (à droite) → « Edit profile »
2. Cliquer sur SSH Keys
3. Générer une clé ssh pour la VM, dans un terminal

```
$ ssh-keygen -t rsa -b 2048 -C "your@emailfrom.git"
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
$ vim .ssh/id_rsa.pub
```

4. Copier-coller la clé publique dans l'éditeur de gitlab
5. Cliquer sur « Add key »

S'il n'est pas déjà existant, il faudra créer un dossier sur le poste de développement.

```
$ mkdir directory_name
```

***Attention** : Pour les stagiaires, les identifiants de connexion à git vous sont fournis par Ange-Marie et/ou Benoit. Ce sont des identifiants uniques.*

Installation de l'environnement du frontend

Pour ce projet il sera nécessaire de faire l'installation de plusieurs programmes et modules sur la VM : node, npm et grunt

Installation de NodeJs

```
$ sudo apt-get update
```

```
$ sudo apt-get install nodejs npm
```

Pour vérifier la version de node et npm faire `node -v` et `npm -v`. Pour information, il est possible d'utiliser nvm pour installer node et choisir la version que l'on veut installer. Pour installer et utiliser nvm, voir : https://snowpact.com/blog/nvm-guide-d-installation

Installer grunt

```
$ sudo npm install -g grunt
```

Pour vérifier la version de node et npm faire `grunt --version`.

Cloner le projet

Depuis le dossier précédemment créer (`cd path_to_directory`, pour s'y rendre en ligne de commande)

```
$ git clone git@gitlab.com:tempora/tempora-frontend
```

Installer le projet

Depuis le dossier tempora-frontend, installer les modules

```
$ npm install
```

Les fonctionnalités

Les fonctionnalités principales de ce module sont :

- Envoyer un message à tous les agendas actifs de la permanence téléphonique.
- Envoyer un message à tous les agendas appartenant aux organisations dont le groupe a été sélectionné.

Les fonctionnalités secondaires de ce module sont :

- Pouvoir personnaliser les informations saisies.
- Permettre à l'utilisateur de sélectionner un message type de la permanence.
- Toutes les fonctionnalités que pourraient proposer les développeurs.

Développement des fonctionnalités

La fonctionnalité étant déjà existante de base, les composants ne seront pas tous à créer. Le projet demande uniquement de l'améliorer.

En amont (avant le développement)

Avant de commencer à travailler sur le développement du projet, il est demandé aux développeurs de réaliser des diagrammes UML, qui permettront de prendre en compte tous les cas possibles et ainsi d'avoir une idée du fonctionnement du projet. Les UML seront à faire valider par le chef de projet. Les diagrammes demandés sont : le diagramme de séquence et le diagramme d'activités.

Une refonte de l'interface étant aussi prévue, une ou plusieurs maquettes seront aussi demandées. Ces dernières devront être validées par Benoit et Ange-Marie.

Pendant le développement

Il est demandé au développeur de mettre à jour régulièrement l'avancement de leurs tâches sur yookkan et de faire des commits, push et pull régulier sur git. Cela permettra au chef de projet et autres membres de l'équipe de suivre l'avancement du projet et d'avoir leur code à jour.

Des tests réguliers en cours de développement sont requis pendant le développement de la fonctionnalité, mais aussi après avoir fait un pull. Cela permettra de vérifier que l'intégration du code récupéré ne casse rien dans le fonctionnement.

En aval (après le développement)

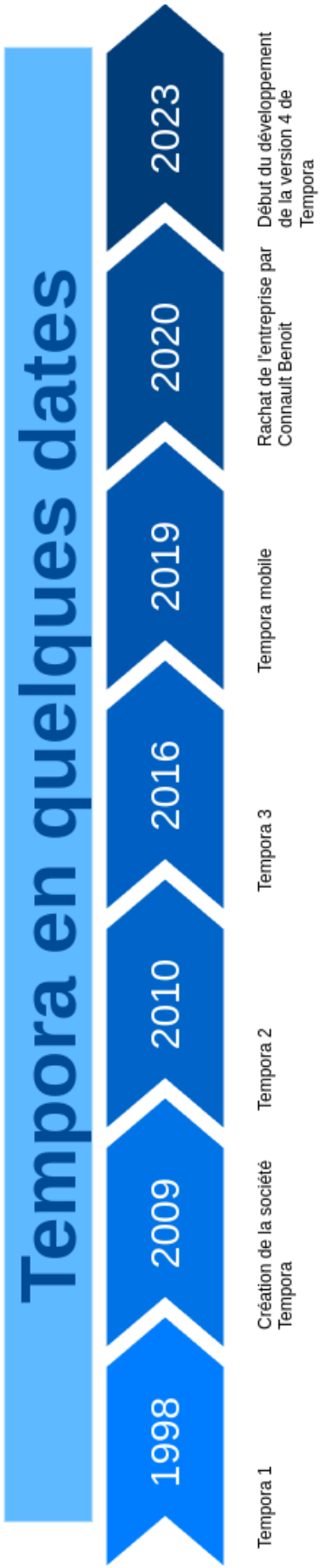
Une fois la tâche terminée, la fonctionnalité et l'application en elle-même doivent être testées dans un premier temps par le développeur puis par un alpha testeur pour être validé. Cette étape de validation permettra de trouver tous les potentiels bugs et de faire une revue du code. Suite à cela, si la validation a été confirmée, les personnes chargées de la tâche pourront commencer une autre itération. Dans le cas contraire, ils pourront corriger les bugs et apporter les modifications remontées, avant de soumettre à une nouvelle validation.

Déroulement du projet

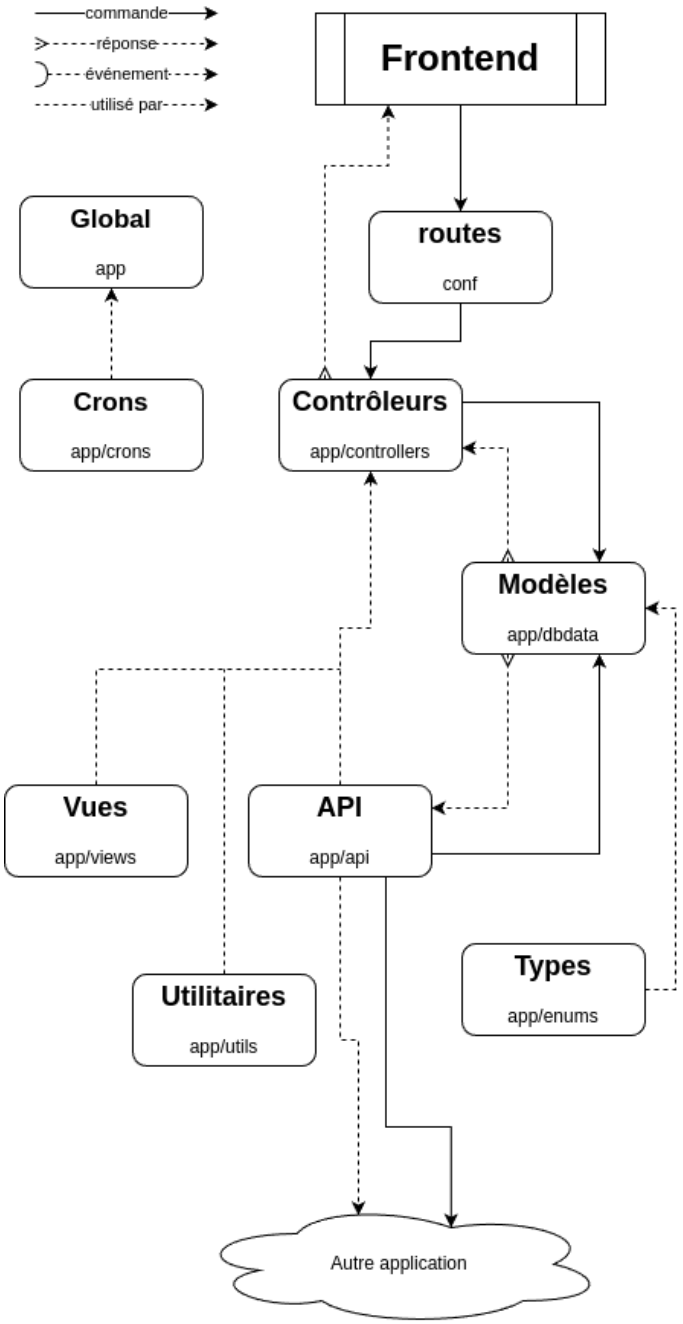
Afin d'avoir un suivi du projet et du stage des points réguliers seront réalisés. Notamment le lundi et le vendredi. La réunion du lundi permet aux différentes parties de dire ce sur quoi ils vont travailler pendant la semaine courante, alors que la réunion

du vendredi quant à elle permet de faire le point sur le travail effectué pendant la semaine passée et de mentionner ce qui n’a pas pu être fini. Cette dernière donne l’occasion aux équipes projet de faire une démo des fonctionnalités finies et ainsi de les faire valider.

Annexe 1 : Frise chronologique de Tempora



Annexe 2 : Architecture du backend



API

Les fichier d'api servent d'interfacage entre Tempora et d'autres application comme google calendar, par exemple.

Modèles

Ils sont l'interfaçage entre les contrôleurs et la base de données. Ils exécutent des requêtes en base de données en fonctions de la demande faites par le contrôleur et traduisent le résultat obtenu en objet fourni au contrôleur.

Contrôleurs

Ils sont au centre des requêtes envoyés par le frontend.

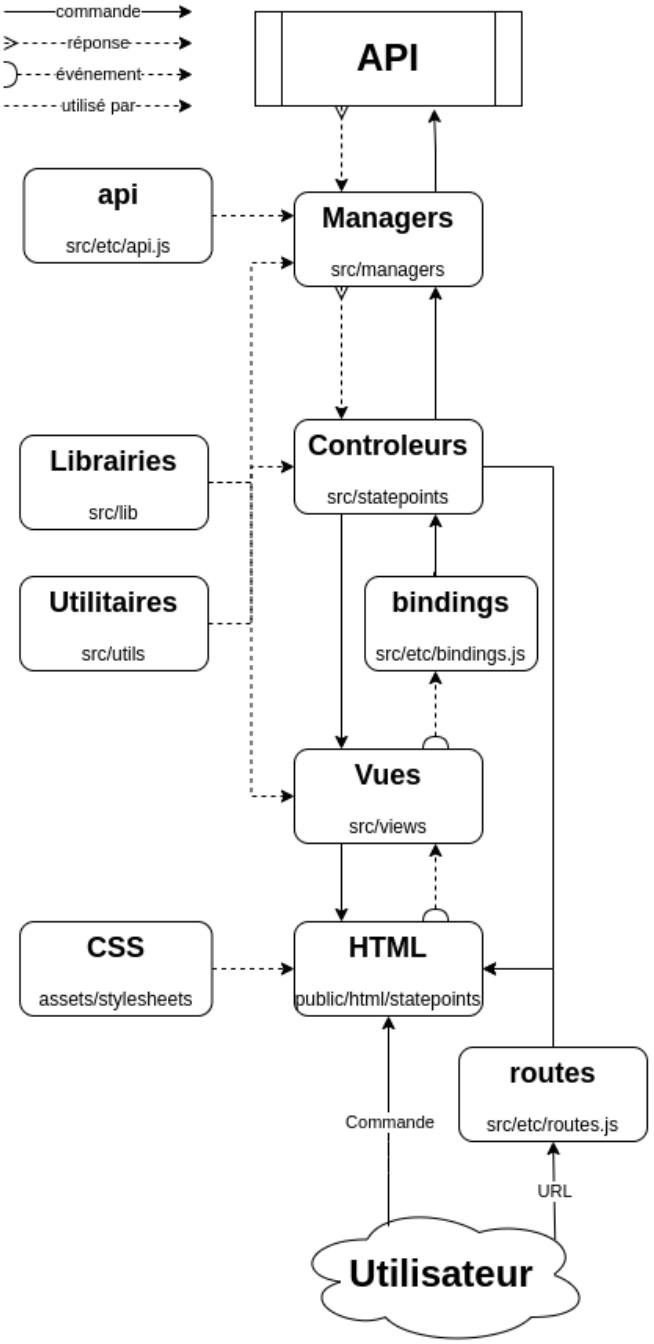
Ils reçoivent les demandes du frontend par le biais des fichier de routes et font appel au dbdata pour les executer dans la base de données.

Ils traitent ensuite les résultats obtenu et renvoi une réponse au frontend.

Vues

Les fichiers de vues sont des fichier HTML, permettant d'envoyer des informations par emails suite à une requête de l'api. Ou bien d'avoir un aperçu avant l'impression.

Annexe 3 : Architecture du Frontend



Managers

Ils sont la passerelle entre les contrôleurs et le serveur API.

Un manager est instancié et appelé par un contrôleur, ce dernier souscrivant à un événement pour attendre la réponse.
Le manager transmet la demande au serveur avec une requête Ajax.
Quand il reçoit la réponse du serveur, il déclenche un événement, que le contrôleur récupère et traite.

L'adresse du serveur est fournie par api.js (en test, penser à modifier api.js pour mettre "local" à la place de "server")

Un manager n'a de référence qu'à lui-même. Il ne "connait" pas les controleurs, vues, etc...

Controleurs

Le coeur du logiciel. Chaque contrôleur gère un aspect/page de Tempora.

Les contrôleurs demandent des données aux managers, traitent les données, demandent d'afficher des données aux vues, et déclenchent le chargement d'une nouvelle page html quand besoin.

Ils reçoivent et traitent également les demandes utilisateurs à travers des bindings.

Un contrôleur "connait" et peut appeler n'importe quel manager et vue, Généralement, un contrôleur est dédié à un seul fichier html.

Vues

Une vue gère l'affichage dynamique d'une page. elle reçoit des données de la part d'un contrôleur, et les affichent en modifiant l'html.

Elle met en place également des événements pour réagir aux commandes de l'utilisateur, qui sont traitées et si besoin transmises au contrôleur en générant un événement, qui sera déclenché et transmis par les bindings.

Généralement, une vue est spécifique à un seul contrôleur, et un seul html.

Une vue ne "connait" que elle-même et l'html lié.

Annexe 4 : Connexion en ssh à une autre machine

```
$ ssh utilisateur@domainedelamachine -p nbPort
```

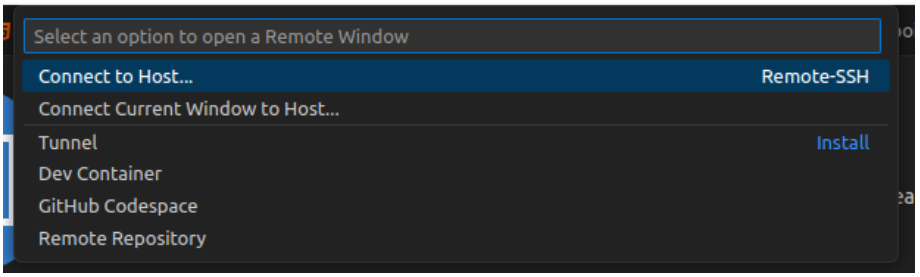
domainedelamachine : peut correspondre à l'adresse ip ou au nom de domaine de la machine sur laquelle il faut se connecter. Pour les vm de Tempora, celui ci est généralement dev.tempora.fr

L'option -p est à mettre lorsque la machine utilise un autre port pour se connecter. Au sein de Tempora, ce dernier est fourni par la personne qui crée la machine, en général Ange-Marie ou Benoit. nbPort, est à remplacer par le numéro de port fourni.

Après avoir exécuté la ligne de commande, un mot de passe pour la machine distante est demandé.

1. Pour ce connecter en ssh depuis vsCode, la ligne de commande fourni ci dessus vous sera utile. Pour initier la connexion ssh depuis vsCode et ainsi pouvoir coder directement sur la machine distante, il faut avoir installer le plugin suivant sur vsCode : Remote - SSH. Puis cliquer sur le bouton qui apparaît en bas à gauche de la fenêtre.

2. Après avoir cliquer dessus, une liste apparaît en haut il faut alors choisir "Connection to Host".



3. cliquer sur "+Add new SSH Host", c'est après cela qu'il faut saisir la ligne de commande précédente.
4. Après avoir entré la ligne de commande, il faut choisir un fichier pour enregistrer la configuration de la connexion ssh. Une fois cette opération validée, reprendre l'étape 1 et 2. Et choisir le domaine affiché, dans la liste.