**\*j’ai l’occasion d’effectuer mon stage d’été chez proxym qui m’acquis de l’expérience et des bonnes connaissances des nv technologies et cette expérience me laisse penser d’être favorisée par rapport aux autres candidats parce que je suis déjà habitué par votre environnement de travail.**

**\*Bien sûr vous pouvez choisir un autre candidat mais je pense que mes qualités et mes compétences que j’ai montré durant mon stage d’été chez proxym l’année dernière ainsi que ma capacité d’apprendre seront un peu bénéfique pour moi.**

**VALEURS DE PROXYM :** GOUT DU CHALLENGE, ESPRIT D’ÉQUIPE, SENS DE L’ÉTHIQUE, TRANSPARENCE, AMÉLIORATION CONTINUE, INNOVATION

**\*Rest :** style d’archi logicielle définissant un ensemble de contrainte à utiliser pour créer des service web

**\*Typeorm :** c’est une orm qui aide à développer tout app qui contient une BD

**\*scrum :** (2 à 4 semaines) est la méthode agile

L'équipe Scrum se compose de :

* Un Scrum Master
* Un Product Owner (ou propriétaire du produit en français)
* Une équipe de développement

Sprint : ensemble des taches à diviser (awel sprint : sprint planing yssir fiha déplacement des taches)

Daily entre l’équipe

**\*Elastic search :** moteur de recherche avancé à base de nosql (recherche complexes) et d’analyse rapide et scalable utilisant la format json

**\*Symphpny**: framework basé sur php

Design pattern : les solutions standards et génériques pour résoudre des prob de conception logiciels

\*singleton : classe qui ne pourra avoir qu’une seule instance

\*MVC : on peut appeler l’instance dans tous les couches de l’applications

\*factory : créer un objet dont le type dépend du contexte : cet objet fait partie d’un ensemble de sous-classes.

\***Angular :** single page application (MVMV)

-component (template html, classe TS : comportement, css selector, ficher test)

-service

Les **observables** prennent en charge la transmission de messages entre les parties de votre application. Ils constituent la technique recommandée pour la gestion des événements, la programmation asynchrone et la gestion de plusieurs valeurs.

**\*Nestjs :**

-contrôleur : traite les demandes, contient des routes(décorateur)

-modele

-service : injecter les dépendances

* Modèle :

SQL VS NOSQL :

|  |  |
| --- | --- |
| SQL(structuré) | NOSQL(non structuré) |
| -BD Relationnel  -Tables -> n lignes de données  -Schéma bien défini  -Evolutive verticalement  -Requêtes complexes  -ACID (Atomicité, cohérence, Isolation, durabilité) | -BD distribuée (ou non relationnel)  -Clé-valeur -> collection de pair clé-valeur  -Schéma dynamique  -Evolutive horizontal -> plus simple  -Requêtes simples  -Préférée pour les ensembles de données volumineux  -Brewers CAP(Cohérence, Disponibilité, Tolérance de partition) |

**\*Cohérence** (*Consistency*) : tous les nœuds du système voient exactement les mêmes données au même moment

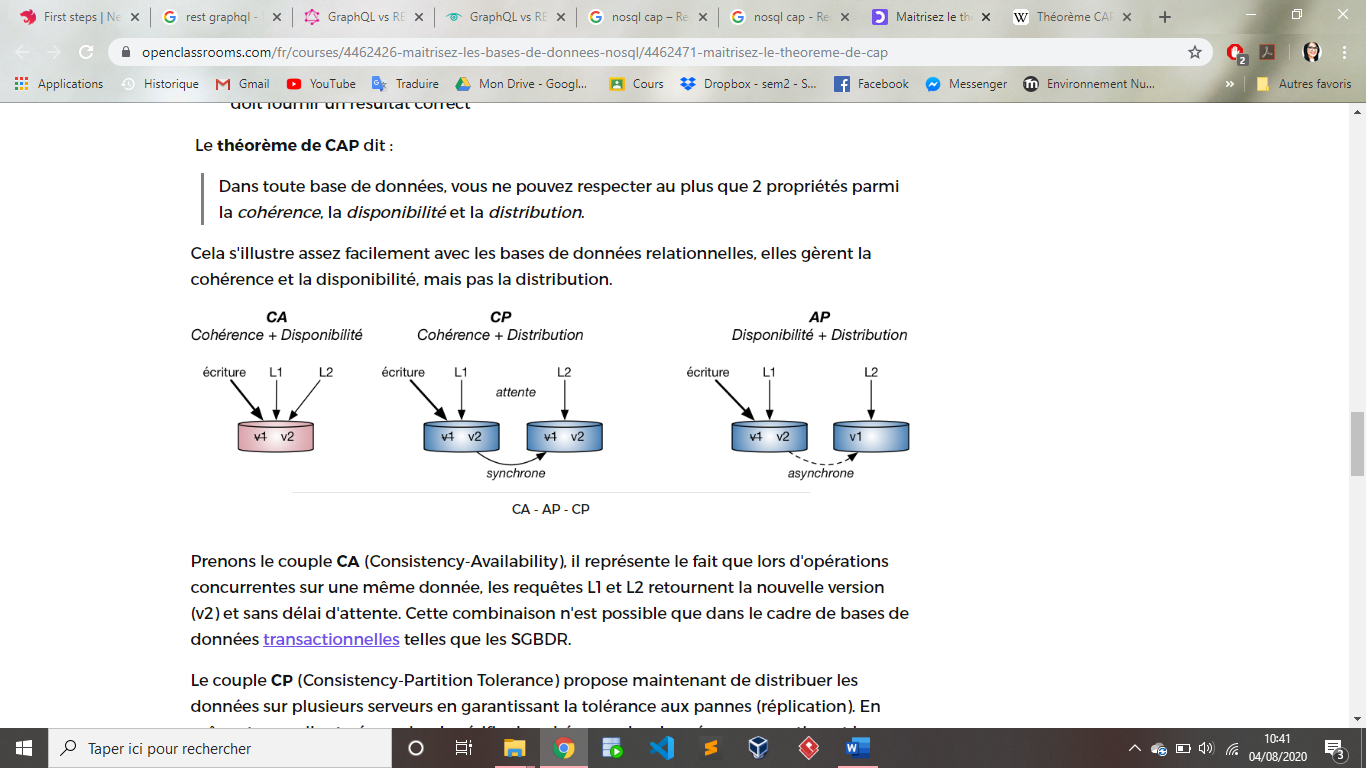
**\* Disponibilité** (*Availability) :* garantie que toutes les requêtes reçoivent une réponse

**\* Tolérance au partitionnement** (Distribution) : Quel que soit le nombre de serveurs, toute requête doit fournir un résultat correct

*Le* **théorème de CAP***dit :*

Dans toute base de données, vous ne pouvez respecter au plus que 2 propriétés parmi la *cohérence*, la *disponibilité* et la *distribution*

\*Consistency-Availability : il représente le fait que lors d'opérations concurrentes sur une même donnée, les requêtes L1 et L2 retournent la nouvelle version et sans délai d'attente.



* Liaison entre back et front : protocole http

Rest vs graphql:

|  |  |
| --- | --- |
| Rest (**Representational State Transfer)** | Graphql (**QL pour Query Language)** |
| * Modèle très structuré basé sur les ressources * Simple en termes de manipulation des données. Chaque entité exposée dans REST est une ressource qui peut être demandée avec des requêtes HTTP * La facilité avec laquelle les données peuvent être « cachées » et la transparence des actions effectuées sur l’API * Exposer une ressource avec toutes les données qui lui sont associées -> toutes ces informations ne seront pas forcément exploitées * [Une API REST comprend un endpoint différent](https://www.lemagit.fr/conseil/6-etapes-pour-booster-la-creation-dAPI) pour chaque ressource, avec des méthodes de requêtage HTTP différentes.   GET /api/articles/1 :  afin de parcourir une liste de chaque article de ID= 1   * Pour créer, mettre à jour ou supprimer un article -> ces opérations standards sont souvent appelées CRUD (Create, Read, Update, Delete). | * Un langage de requête pour les API plus flexible * L’application et sa structure de donnée font appel à de nombreuses requêtes (requêtes imbriquées) * Plus programmatique et permettre au client de requêter les seules informations souhaitées au final * Permettra au client d’interagir avec les nouvelles données d’une façon plus cohérente * Permet de centraliser les requêtes dans un seul point d’entrée, les opérations à effectuer sont passées dans la requête elle-même. * GraphQL utilise uniquement deux opérations (query et mutation) pour consulter ou mettre à jour les données. * L’un des principaux inconvénients est la gestion du cache |

* Liaison entre front et modèle : ORM

SQL : Sequelize / TypeORM

|  |  |
| --- | --- |
| Sequelize | TypeORM |
| * Il offre une prise en charge solide des transactions, des relations, un chargement rapide et paresseux, une réplication en lecture et plus encore * Les modèles sont l'essence même de Sequelize. Un modèle est une abstraction qui représente une table dans votre base de données * Très simple à utiliser et ainsi de posséder une courbe d'apprentissage extrêmement rapide. * Offre nativement une connectivité aux principales bases de données   Open source telles que PostgreSQL, MySQL, MariaDB, SQLite et MSSQL.   * Il se chargera de créer les tables automatiquement ou bien l'utiliser pour une bd déjà existante * Sequelize dispose d’un large éventail de fonctionnalités qui couvre : les requêtes, les étendues, les relations, les transactions, les requêtes brutes, les migrations, la réplication en lecture, etc. * Sequelize est basé sur des promesses, ce qui facilite la gestion des fonctions asynchrones et des exceptions. | * est un [ORM](https://en.wikipedia.org/wiki/Object-relational_mapping) qui peut fonctionner sur les plates-formes NodeJS, Browser, Cordova, PhoneGap, Ionic, React Native, NativeScript, Expo et Electron * TypeORM prend en charge à la fois [les](https://typeorm.io/#/active-record-data-mapper/what-is-the-active-record-pattern) modèles [Active Record](https://typeorm.io/#/active-record-data-mapper/what-is-the-active-record-pattern) et [Data Mapper](https://typeorm.io/#/active-record-data-mapper/what-is-the-data-mapper-pattern) , contrairement à tous les autres ORM JavaScript existants, ce qui signifie que vous pouvez écrire des applications de haute qualité, faiblement couplées, évolutives et maintenables de la manière la plus productive. * peut être utilisé avec TypeScript et JavaScript (ES5, ES6, ES7, ES8) * Comme il est écrit en TypeScript, il fonctionne plutôt bien avec le framework Nest. |