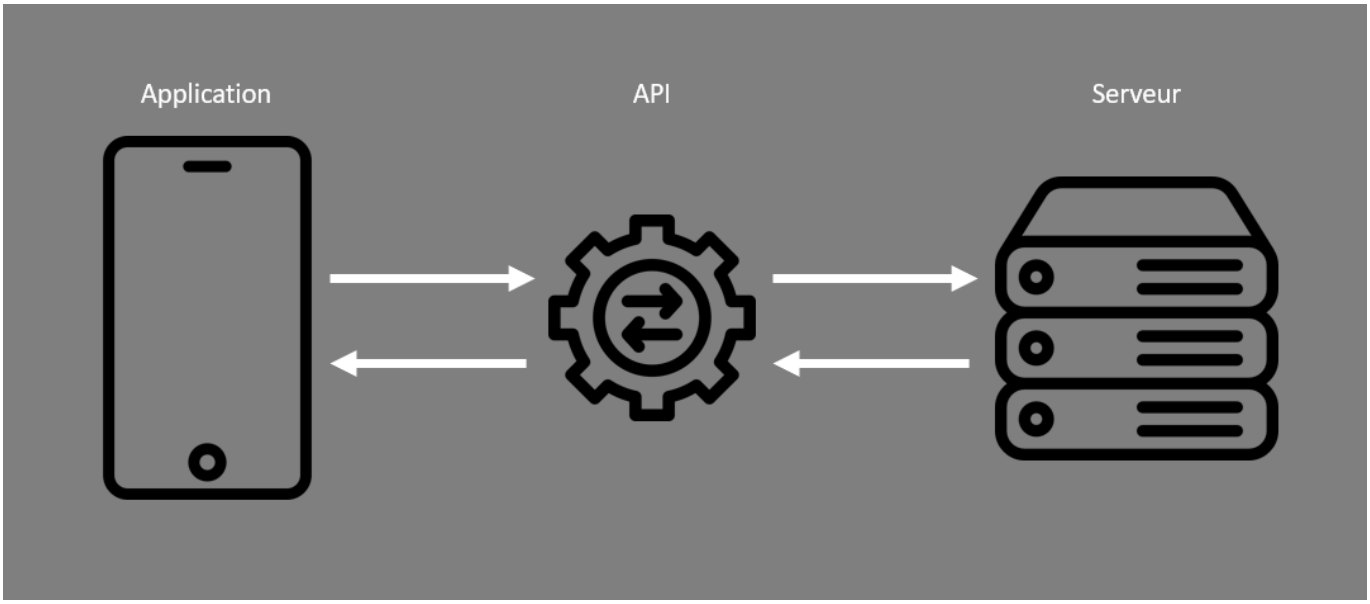
Atelier développement d'une application mobile

Dans le cadre de la matière de développement mobile nous avons à concevoir et développer une API (Application Programmation Interface) et une application mobile de gestion de festival. Tout d'abord, il est nécessaire de préciser la définition du terme API que je ne connaissais pas avant d'avoir eu ce projet. Voici un schéma qui permet de visualiser où se trouve une API :



Le mot "interface" présent dans l'acronyme API indique bien l'objectif de celle-ci qui a pour but de faire de l'échange mutuel de données entre le serveur et l'application. Pour présenter ce projet, je vais procéder en plusieurs étapes. D'abord, je vais présenter les différents choix techniques que j'ai fait. Puis, j'évoquerai l'application et son fonctionnement. Je vais ensuite expliquer le fonctionnement de mon API. En outre, j'aborderai la sécurité de l'application. Enfin, je révélerai les différentes futurs évolutions de mon projet.

Choix techniques

Etant donné que nous avons à développer une application mobile, nous devons donc faire le choix d'un langage qui permet de la développer. Malgré le fait que j'ai pu pratiquer le swift lors de mon cursus universitaire, je trouve que ce langage est trop ciblé puisqu'il ne permet que de développer une application pour les téléphones ayant le système d'exploitation iOs. J'ai donc choisi le Flutter qui nous a été présenté assez récemment lors du début de l'année pour plusieurs raisons :

- Le Flutter permet de déployer son application mobile sur n'importe quelle plateforme que ce soit pour un téléphone Android ou iOs. Le plus grand nombre d'utilisateur que cette

application peut toucher, permet donc une plus grande visibilité sur l'application.

- Il s'agit du **framework le plus utilisé par les développeurs** depuis cette année en dépassant le React Native développé par l'entreprise Facebook renommée Meta recemment. Le React Native paru dès 2015 est pourtant très utilisé par des entreprises influentes comme Walmart, Instagram, Airbnb, Tesla et bien d'autres.

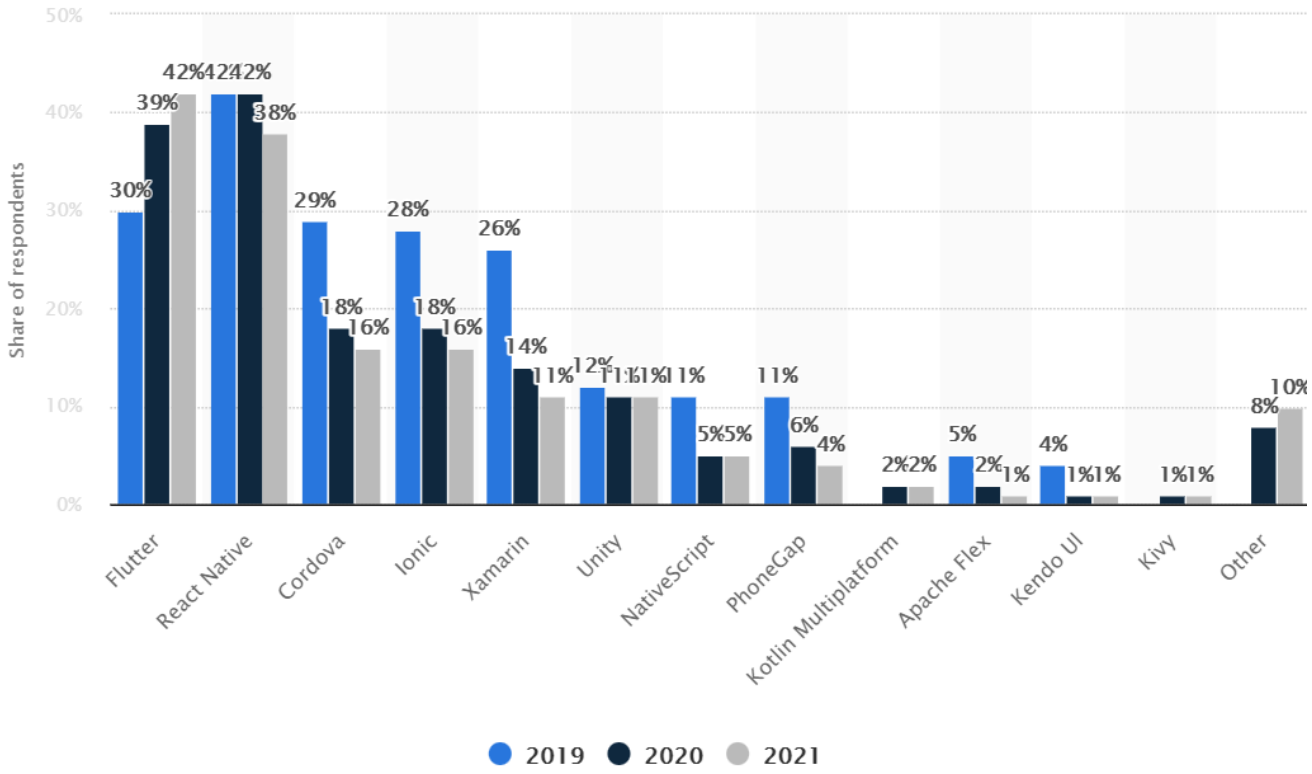


Schéma illustrant un pourcentage de développeurs utilisant le framework Flutter depuis 2019
source : Cross-platform mobile frameworks used by global developers 2021 | Statista

Ce langage possède donc une **grande communauté** ce qui induit des **forums d'aides très actifs** qui sont très utiles lorsque l'on rencontre une difficulté particulière.

- Je trouve que le Flutter est **assez intuitif** et **simple à apprendre**.

Comme le précise le projet, je dois également développer une API. Mes recherches aboutissent sur le concept REST. Cela répond totalement au besoin demandé par le projet. En effet, la demande se porte sur l'ajout, la modification et la suppression des artistes et utilisateurs. Tout ceci correspond au modèle CRUD ("Create Read Update Delete" en français "créer lire modifier et supprimer").

REST signifie "REpresentational State Transfer", il a été proposé par Roy Fielding en 2000 comme étant une approche architecturale de la conception de services web. Voici une citation qui illustre très bien le concept REST : *"L'API REST est au développeur ce que l'interface

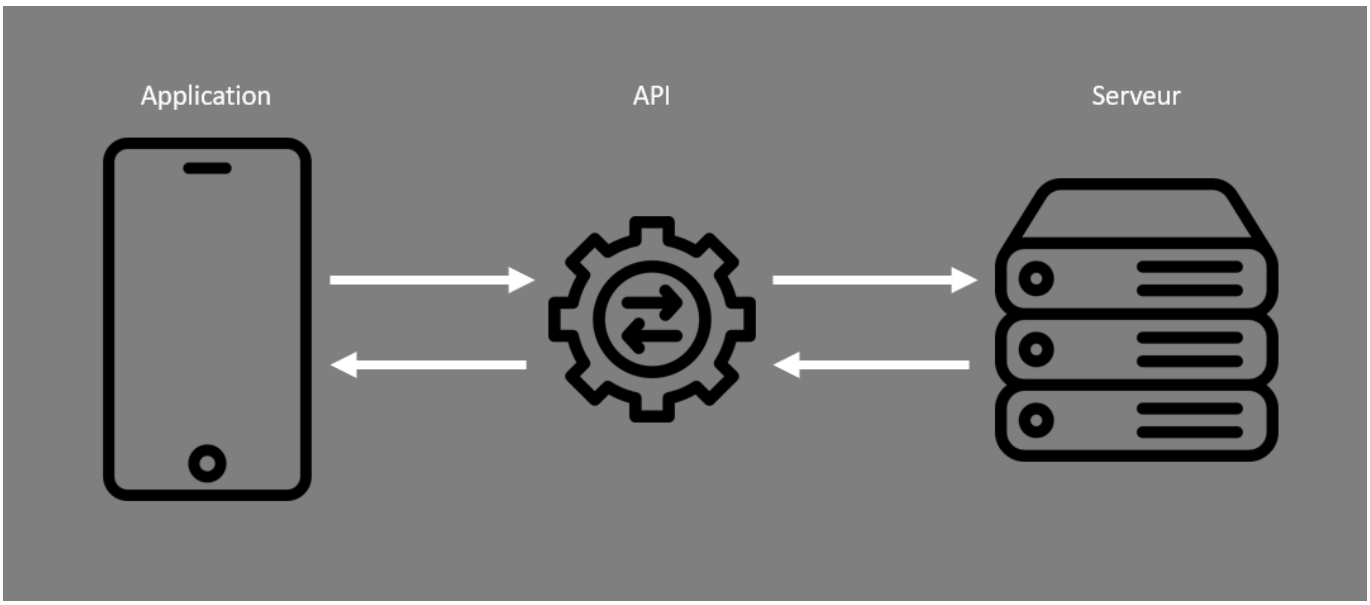
utilisateur est à l'utilisateur final, une méthode de communication avec un serveur."* Les bonnes pratiques à suivre pour développer des APIs REST – Gekko

Les services web utilisant le style architectural REST sont nommés services web RESTful. Ils sont donc normés et uniformisés dans le but d'accéder aux ressources web et de les manipuler. Un système RESTful se base sur un concept assez simple qui est le suivant : un client va demander une ressource et le serveur va posséder ces ressources.

Afin que notre API respecte les normes et devienne RESTful, nous allons devoir respecter les six contraintes qui sont les suivantes :

- **Architecture Client/Serveur** : avec des requêtes http permettant la liaison client/serveur. Les applications clients et serveurs doivent être totalement indépendantes.
- **Stateless** (sans-état) : les données ne sont pas stockées sur le serveurs mais sur le client.
- **Cacheable** (cachable) : la mise en cache des informations permet d'améliorer la performance du fait que si on fait une requête "get" deux fois la deuxième fois la requête arrivera plus vite du fait qu'elle est mise en cache.
- **Layered system** (système à plusieurs couches) : on peut déployer les API sur plusieurs serveurs avec des fonctionnalités différentes. Un serveur peut servir seulement de connexion. Alors qu'un autre peut uniquement stocker des données etc. Les serveurs et API forment donc des couches.
- **Uniform Interface** (Interface uniforme) : chaque ressource doit pouvoir être partagée via une requête http.
- **Code on demand** (Code sur demande) : il s'agit d'une contrainte facultative mais les ressources partagées dans les requêtes http sont souvent statiques. Cependant, on peut également partager du code exécutable.

Ce concept est donc très associé au web donc au protocole HTTP (HyperText Transfer Protocol) mais il n'y est pas forcément lié. Je vais utiliser ce dernier protocole pour le travail qui m'est demandé. Il va m'être utile afin de faire le lien entre l'application mobile et l'API ainsi que le serveur. Vous pouvez l'observer sur le schéma ci-dessous avec les flèches blanches qui correspondent aux requêtes.



D'après mes différentes recherches, beaucoup de langages de programmation permettent le développement d'une API. Je n'ai jamais pratiqué le développement d'une API donc j'ai choisi de développer celle-ci dans un langage que je connaissais. Le langage de programmation que je connais le plus et où je suis le plus à l'aise est le Java et nous le pratiquons en cours.

J'ai découvert le framework Spring en cherchant et en suivant les conseils de certains de mes camarades de classe l'ayant déjà pratiqué. Le sujet de ce projet évoque la sécurité de la connexion sur l'application. Spring propose le framework Spring Security permettant d'effectuer la sécurisation de l'authentification que je vais utiliser. On peut également utiliser les token jwt dans ce framework permettant également d'assurer la sécurité de l'application. Je présenterais plus en détail ceci dans la partie sécurité de ce document. Afin de tester le bon fonctionnement de l'API j'ai utilisé l'outil Swagger qui est également disponible sur cette API.

Fonctionnement de l'application

Fonctionnement de l'API

Sécurité de l'application

Etant donné que je ne suis pas un expert en cybersécurité et que c'est la première fois que je développe une application mobile ainsi qu'une API je ne pense pas que cette application soit très sécurisée. J'ai utilisé le framework Spring Security et surtout les JSON Web Tokens (jwt). Les jwt sont des tokens d'accès. Ils vont permettre à notre administrateur de l'application mobile d'accéder aux données présentes sur la base de données via l'API de manière sécurisé.

Les futurs évolutions souhaitées

- Utiliser une api open source vue en cours nommée Key Cloak. Il s'agit d'une API développée par la société Red Hat permettant une meilleure sécurité pour la connexion. En effet, je n'ai jamais développé d'API auparavant et je ne connais pas les différentes failles de sécurité existantes et comment sécuriser mon application de la meilleure manière possible. Red Hat est très connu dans le monde de l'informatique puisque c'est le premier éditeur open source mondial. Cette entreprise est donc très compétente dans beaucoup de domaines ce qui lui permet de sécuriser l'API de connexion qu'elle met à disposition avec les dernières failles de sécurité connues, de la sécuriser de la meilleure manière possible avec des développeurs expérimentés en cybersécurité. Key Cloak permet également une gestion du trafic qui peut être utile pour un festival quand des milliers de personnes se rendent sur l'application en même temps avec l'outil Kubernetess par exemple.
- Un plan du festival interractif permettant que lorsqu'on clique sur une zone, cela nous renvoie vers les détails de cette zone avec ses différents stands, le planning des artistes prévus etc.
- Un champ texte mutable sur l'application en haut permettant de voir l'artiste en qui se produit actuellement sur une des scènes du festival.
- Une application permettant de gérer plusieurs festivals.
- Une base de données distante qui n'est pas stockée localement sur mon ordinateur.

Dépot de code :

Vous pouvez trouver le code de mon API Spring Boot à l'adresse suivante :

GitHub - Ludiphil/projetFestival

Vous pouvez trouver le code de mon application mobile Flutter à l'adresse suivante :

GitHub - Ludiphil/festival_app