MELLAH Rayanne
GUNES Omer
PERON Romain
PARCOLLET Noé

Rapport de veille technologique

1. Introduction aux solutions :

Pour réaliser le projet MathsALaMaison plusieurs options s'offre à nous. Le premier est de réaliser un site internet au format mobile pour accéder au jeu (le jeu sera alors sous forme de webapp). Le second est de réaliser une application natives pour chaque plateforme (iOS et Android). Le troisième est de réaliser une application natives pour les deux plateformes à la fois.

Pour la première option, les technologies pour créer l'application sont **HTML/CSS ou React** pour le front-end et **PHP, JavaScript, MySQL, OracleDatabase** (toute autre technologie de base de données).

Pour la deuxième option, il faut créer deux applications distincte, une pour la plateforme iOS et l'autre pour la plateforme Android. Les deux utilisant des technologies différentes. Pour la plateforme IOS, le côté client est développé via du **Swift** ou de l'**Objective-C** mais il commence à être obsolètes. Pour le côté serveur on peut utilisé n'importe quelle technologie de base de données comme **MySQL, OracleDatabase**. Pour la plateforme Android, le côté client peut être developpé avec multiple langages tel que **Java, Kotlin, C, C++, C#**. Pour le côté serveur on peut utilisé n'importe quelle technologies de base de données comme **MySQL ou OracleDatabase.**

Pour la dernière option, il faut créer une seul application qui va être compatible au deux plateformes. La technologie qui peut être utilisé à cette usage est le **ReactNative**. Les technologies pour le backend sont **TypeScript, JavaScript** et n'importe quelle technologies de base de données comme **MySQL ou OracleDatabase.**

2. Critères :

Il faut prendre des caractéristiques communes à chaque langage pour trouver lequel est le meilleur objectivement.

Pour la liste des critères on peut noter :

- Apprentissage du langage
- Compatibilité entre les différentes plateformes
- Maintenabilité
- Durée de vie
- Coût

L'**apprentissage** du langage/framework doit obligatoirement être pris en compte, on ne va pas mettre le même temps a prendre en main TypeScript et le C, ce qui entrainera des retard dans le projet.

La **compatibilité** du langage/framework est important car si un langage est compatible à plus de plateforme. Cela évite de développer plusieurs fois pour que ça soit compatible à tout le monde.

La **maintenabilité** il faut que le code reste maintenable utilisé un fichier qui contient du ReactNative qui contient la vue et contrôleur deviendra vite très compliqué à gérer. Le mieux est de séparé les différentes parties du code.

La **durée de vie,** utilisé un framework qui vient de sortir peut être bénéfique à long terme car on il sera en pleine émergence. Cependant cela peut aussi poser des difficultés pour trouver des packages et d'outils pour le développement. A l'inverse, utilisé un langage obsolètes posera également des problème. Bien que beaucoup de ressources soit disponible, ce langage est voué à disparaitre. Le mieux est de choisir une technologies qui se trouve en pleine équilibre entre maturité et les ressources disponible, comme Java.

Le **coût** d'utilisation d'une technologie peut vite devenir élevé, sur le plan financier mais aussi sur le temps. Par exemple certaine technologie notamment le développement sur la plateforme iOS en natives nécessite des équipement comme un Mac et un abonnement. De plus si la technologie est assez complexes à prendre en main cela peut vite devenir long pour développer l'application.

3. Comparaison :

Dans cette partie nous allons comparée les différentes technologies existantes pour le développement du projet du côté client.

Dans le tableau nous allons mettre des notes allant de 1 à 10 pour chaque langage/framework en fonction des critères qu'on a défini dans la deuxième partie.

Critères	HTML	React	Java	ReactNative	Kotlin	Swift
Apprentissage	1	8	2	8	7	6
Compatibilité	10	10	4	10	5	5
Maintenabilité	2	7	8	6	6	6
Durée de vie	3	6	3	5	4	4
Coût	1	4	4	4	7	10

Apprentissage :

- HTML : Très facile à apprendre, avec une courbe d'apprentissage faible
- React : Framework de JavaScript assez populaire, mais complexe à maitriser
- Java : Langage assez robuste et documenté. Les bases sont simples, mais compliqué pour les grandes applications.
- ReactNative : Similaire à React, mais avec des technologies propres au mobile.
- Kotlin : Langage nouveau, plus facile que le Java.
- Swift : Assez facile mais nécessite un écosystème Apple.

Compatibilité :

- HTML : Compatible avec toutes les plateformes web.
- React : Compatible avec les applications web et de nombreuses plateformes
- Java : Orienté pour le back-end et le développement Android, mais on peut comme même faire du Front-end même si c'est plus compliqué que les autres.
- ReactNative : Compatible avec le développement mobile, permettant de créer une application avec un seul code.
- Kotlin : Orientée pour le développement Android, donc assez limité.
- Swift : Orientée pour le développement sur iOS, donc assez limité.

Maintenabilité :

- HTML : Très simple à maintenir, car c'est un langage très simple
- React : Devient assez difficile a gérer avec les dépendances.
- Java : Assez robuste, mais assez compliqué à maintenir sur de gros projets
- ReactNative : Similaire à React mais on doit gérer les différences enter les plateformes.
- Kotlin : Assez maintenable, mais compliqué sur de gros projets.
- Swift : Assez maintenable, mais compliqué sur de gros projets.

Durée de vie :

- HTML : Assez ancien, mais toujours autant utilisé donc il risque pas de devenir obsolète.
- React : Utilisé beaucoup donc il y a beaucoup de ressources sur internet.
- Java : Langage beaucoup utilisé avec beaucoup de ressource et risque pas de devenir obsolète.
- ReactNative : Similaire à React, pas de risque de devenir obsolete tant qu'il reste supporté par iOS et Android.
- Kotlin : Supporté par Google et Android, donc promet une durée de vie élevée
- Swift : Tant que l'écosystème Apple et en place il ne risque pas de devenir obsolète.

Coût :

- HTML : Très peu couteux
- React : Très peu couteux à mettre en place
- Java : Très peu couteux à mettre en place
- ReactNative : Plus couteux à mettre en place (il faut des abonnement pour publier les projets pour iOS et Android)
- Kotlin : Couteux en raison d'un abonnement pour publier le projet
- Swift : Couteux également en raison d'un abonnement nécessaire pour publier le projet.

Dans le tableau suivant nous allons comparée les technologies du back-end.

Critères	TypeScript	JavaScript	MySQL	OracleSQL
Apprentissage	Plus facile que le JS	Assez compliqué à apprendre avec des fonctions asynchrones	Facile à apprendre	Plus difficile que le MySQL
Compatibilité	Compatible avec le JavaScript	Compatible avec tous les navigateurs web	Fonctionne avec plusieurs framework et langage.	Compatibilité limité avec les framework ou langage
Maintenabilité	Typage statique donc plus simple	Moins maintenable à cause du typage dynamique	Structure de la bdd claire.	Necessite plus d'effort que MySQL
Durée de vie	Utilisé énormément grâce aux typages	Utilisé par beaucoup de monde	Utilisé par énormément de monde	Utilisé par beaucoup de banque et de gouvernrement
Coût	Open Source	Open Source	Gratuit	Coût de licence

4. Conclusion :

Pour le projet MathsALaMaison nous avons décidé d'utiliser les technologies suivant :

ReactNative pour le front-end pour avoir une application compatible avec tous les smartphones et tablettes comme demandé par la cliente. Pour le back-end de l'application nous allons d'abord utiliser du TypeScript pour séparer les fonctions de contrôle et les vues. Nous allons également utiliser une backend API qui va permettre de faire les requêtes à un serveur de base de données externalisé qui va permettre de centraliser toutes les données. Pour le système de stockage des données on a décidé d'utiliser MySQL sur un serveur externe pour tout centralisé.