Angelo ALIBERT
Etienne REYGNER

INFO5
17/03/2023

Rapport final de projet

Projet : Adaptation d'une plateforme web à un protocole expérimental sur les entretiens vidéos asynchrones

Sujet proposé par : Christelle Martin-Lacroux, CERAG/UGA,
Sybille Caffiau, LIG/UGA,
Sophie Dupuy-Chessa, LIG/UGA

Table des matières

Contexte **3**
Présentation du sujet 3
Une étude menée par le LIG et le CERAG 4
Cahier des charges 4
Maquette IHM 4
Modèle de base de données 5
Technologies employées **5**
Architecture technique **6**
Réalisations techniques **6**
Front-end 6
Back-end 11
Gestion de projet **14**
Méthodologie AGILE 14
Weekly meetings 14
Evaluation des contraintes 14
Evaluation des risques 14
Trello 15
Rôles des membres 15
Outils de collaboration **16**
Métriques logiciels **16**
Lignes de code & commits 16
Langages 16
Conclusion **17**
Démonstration **17**
Annexes **18**
Annexe 1 : Modèle de base de données 18
Annexe 2 : Maquette de l'interface utilisateur 18

Contexte

Présentation du sujet

Avec l'évolution des technologies numériques et la démocratisation du travail à distance, de plus en plus d'entreprises ont recours à l'utilisation du recrutement à distance (on parle d'entretiens vidéo asynchrones).

Le principe est le suivant : les candidats reçoivent une invitation d'une entreprise à se connecter à un site ; ils reçoivent ensuite une liste de questions à préparer et enregistrent leurs réponses. Les réponses des candidats sont ensuite envoyées et évaluées par un système d'analyse automatique intégrant de l'IA et/ou par un recruteur humain.

Le recrutement vidéo asynchrone offre de nombreux avantages aussi bien pour les entreprises que les candidats tels que :

- La flexibilité : les candidats peuvent répondre aux questions à leur rythme, en ayant le temps de se préparer. Aussi, cela leur permet de passer l'entretien au moment souhaité en fonction de leur emploi du temps, sans avoir à se coordonner avec un recruteur.
- Gain de temps et d'argent : les frais de déplacement sont évités et les recruteurs gagnent un temps précieux en pouvant analyser les réponses quand ils le souhaitent.
- Evaluation plus approfondie : les recruteurs peuvent analyser les réponses plusieurs fois et poser des questions plus approfondies.
- Amélioration de l'expérience candidat : offre aux candidats un moyen plus pratique de postuler à des postes et en leur permettant de se sentir plus à l'aise et moins stressés lorsqu'ils répondent aux questions d'entrevue.
- Image de marque : les plateformes personnalisées peuvent affirmer l'image de marque d'une entreprise.

Cependant il est important de noter que ce mode de recrutement peut poser certains problèmes d'ordre technique (par exemple si le candidat dispose d'une mauvaise connexion à internet). On peut également penser à l'aspect éthique de l'utilisation d'une IA et le manque d'interaction avec un recruteur humain dans l'analyse des candidatures.

Une étude menée par le LIG et le CERAG

Deux laboratoires de l'UGA s'intéressent à ce mode de recrutement :

- Le CERAG (Centre d'Études et de Recherches Appliquées à la Gestion) souhaite observer le comportement d'une centaine d'invidus placés dans une situation de candidature par l'intermédiaire d'un questionnaire.
- Le LIG (Laboratoire d'Informatique de Grenoble), souhaite étudier les données produites par les interactions nécessaires pour ce mode de recrutement telles que des vidéos, questionnaires et interactions sur l'interface.

Les avantages attendus de cette étude sont, à terme, d'obtenir une meilleure compréhension des attitudes et des comportements des candidats exposés à la méthode des entretiens vidéo différés.

Cahier des charges

Afin de mener à bien cette étude, un protocole expérimental détaillant tous les aspects de l'étude a été élaboré au préalable par le CERAG. Ce dernier joue le rôle de cahier des charges dans notre cas.

Maquette IHM

Les avantages de ce type de maquettes sont nombreux. Elles permettent notamment de créer une représentation visuelle quasiment fidèle du site web final, ce qui permet aux clients de mieux comprendre l'apparence et le fonctionnement du site avant même que le développement ne commence. In fine, la communication avec le client est améliorée et le risque de malentendu est réduit.

Afin de respecter au mieux les attentes du client, nous avons donc commencé par réaliser une maquette d'IHM (Interface Homme-Machine) contenant de manière présice les pages de la plateforme, la position des éléments, les boutons, formulaires, textes... ainsi que le bon enchaînement des pages en se basant sur la plateforme que nous avions à adapter.

Pour réaliser notre maquette d'IHM, nous avons utilisé la plateforme Whimsical. Elle permet du travail collaboratif en temps réel et propose des outils de création adaptés tels que des widgets (boutons, formulaires, onglets) qui sont des éléments que l'on retrouve sur un site web. Notre maquette est consultable via ce lien.

Grâce à cette maquette nous avons pu montrer et faire valider les propositions. Une fois que la maquette a été validée, nous avons pu nous lancer sereinement dans la phase de développement.

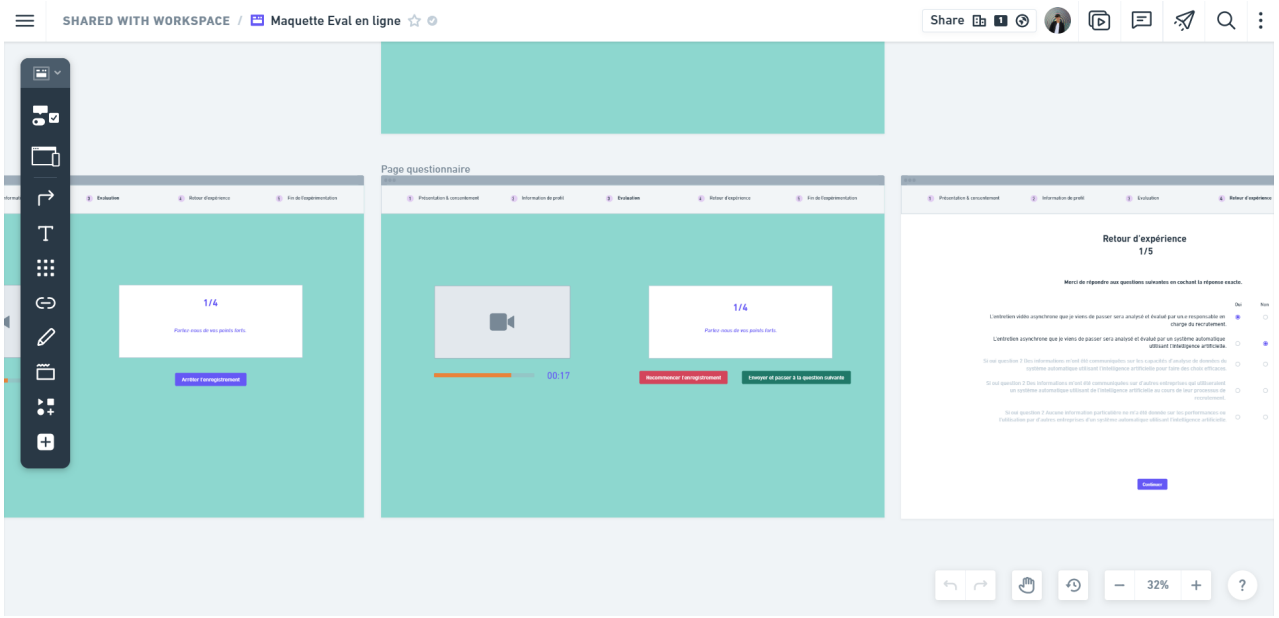


Figure : Extrait d'une page de la maquette sur la plateforme Whimsical.

Modèle de base de données

Nous avons ensuite défini un modèle de base de données en nous basant sur celui déjà existant. Il a fallu mettre en œuvre des solutions pour gérer l'anonymat des données selon les critères du RGPD et enregistrer toutes les données utiles au protocole.

Vous trouverez en annexe le diagramme UML de la base de données. (voir Annexe 1)

Une descriptions plus précise de notre travail sur la base de données est faite plus bas dans la partie "Réalisations techniques / Back-end"

Technologies employées

Pour notre projet, nous avons utilisé les technologies déjà implémentées dans le projet. La réalisation du front-end est faite avec Angular. Le serveur qui récupère les données et stocke les fichiers de consentement et les vidéos est réalisé avec ExpressJS. Les données sont stockées avec SQLite. Enfin, pour l'API nous avons utilisé GraphQL qui permet de simplifier et d'optimiser la récupération des données. Ensemble, ces technologies fournissent une pile flexible pour construire des applications web modernes.

Architecture technique

L'application est basée sur une architecture client-serveur. Les serveurs du LIG hébergent le back-end de l'application chargé de stocker les informations sur le profil du candidat, une copie du formulaire de consentement, ses réponses lors de l'entretien, ainsi que ses interactions avec la plateforme (logs de clics...).

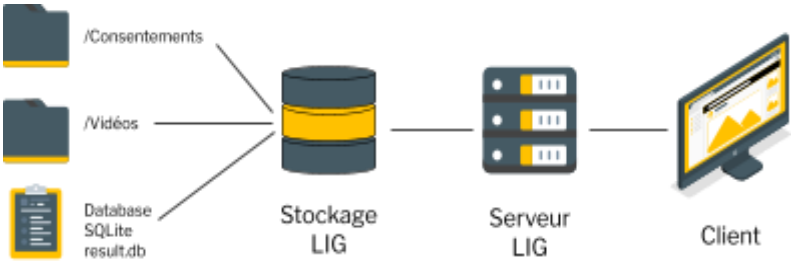


Figure : Schéma de l'architecture de notre projet.

Réalisations techniques

Front-end

Pour ce projet nous sommes partis d'une base déjà existante d'une plateforme d'étude issue de précédents travaux menés au LIG ayant pour but de faire évaluer l'interface d'un site web à des centaines de candidats. Celle-ci enregistrait les interactions de l'utilisateur (clavier/souris) ainsi que la vidéo de l'utilisateur (pour le déplacement des yeux). En partant de ce projet nous avions les fonctionnalités utiles pour la réalisation finale. Des parties sont communes aux deux projets mais ont été adaptées au protocole que nous avons mis en place.

1. Consentement

Figure : Capture d'écran de la plateforme : page consentement.

Pour pouvoir participer à l'étude, l'utilisateur doit donner son consentement en remplissant un formulaire. Ce formulaire est stocké sur le serveur au format pdf et dans la base de données. Une copie du pdf est téléchargée sur la machine de l'utilisateur.
Cette partie est reprise de l'ancien projet, les textes expliquant l'étude ont été modifiés afin de correspondre aux besoins du protocole.

2. Profil

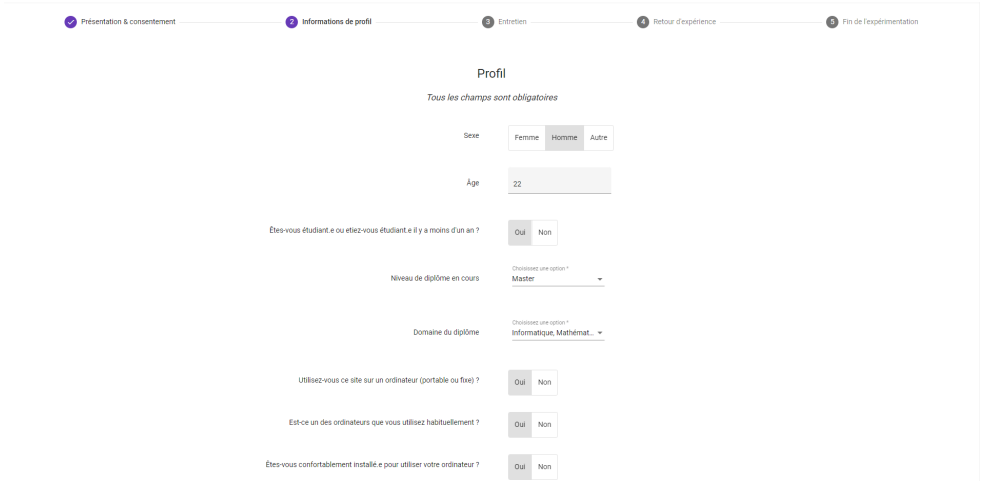


Figure : Capture d'écran de la plateforme : informations de profil.

L'utilisateur doit ensuite compléter son profil afin de voir si celui-ci est compatible avec l'étude menée. Si l'utilisateur ne correspond pas aux critères, celui-ci est redirigé vers la page de remerciements. Des modifications ont également été réalisées sur la partie de complémentation du profil. En ajoutant de nouvelles questions et de nouveaux critères d'exclusion de l'enquête.

3. Contexte

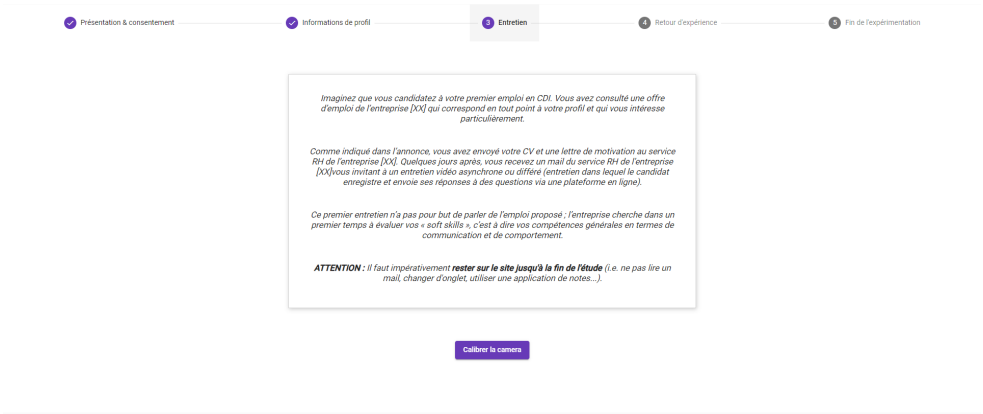


Figure : Capture d'écran de la plateforme : mise en situation.

Cette page indique à l'utilisateur le contexte de l'étude et le rôle qu'il doit prendre pendant celle-ci. Cette mise en situation permet de faire une transition vers l'étude. Cette page a été créée spécialement comme indiqué dans la maquette.

4. Calibrage caméra

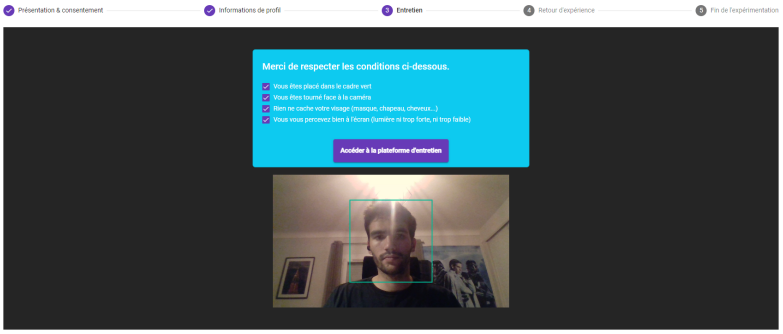


Figure : Capture d'écran de la plateforme : calibrage caméra.

A cette étape, la caméra du navigateur s'allume et l'utilisateur à un aperçu de ce qui va être enregistré. Des consignes lui sont également données afin de ne pas obstruer l'enregistrement. La plateforme se met automatiquement en plein écran à cette étape de l'étude. Cette page a été légèrement modifiée.

5. Mise en situation

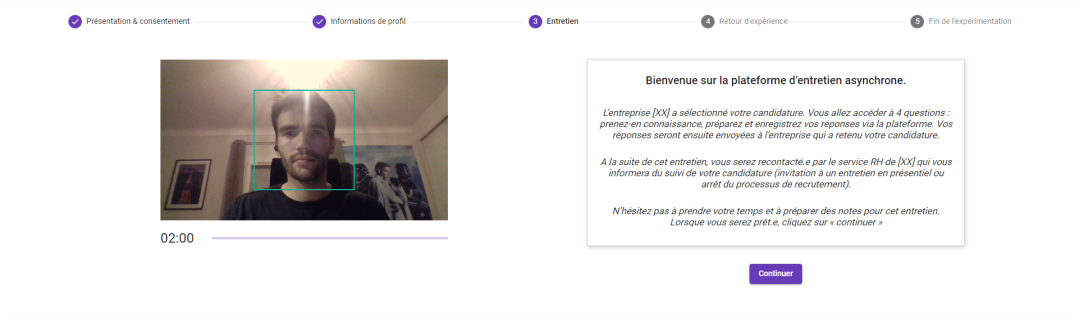


Figure : Capture d'écran de la plateforme : début de l'entretien.

Une fois le calibrage terminé, un texte apparaît tel qu'en situation réelle de recrutement. Les pages de l'entretien ont toutes été conçues spécialement pour le protocole actuel. Ces pages suivent la même structure pour garder une cohérence. L'affichage de la caméra dans le navigateur a été revu car l'ancienne version pouvait poser des problèmes d'affichage.

6. Condition

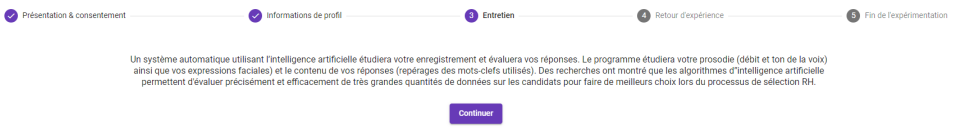


Figure : Capture d'écran de la plateforme : affichage des conditions.

Durant l'étude, l'utilisateur est associé aléatoirement à une "condition". Il existe 6 conditions. Parmi elles, 3 indiquent à l'utilisateur avant le début de l'entretien que la vidéo sera examinée par un utilisateur ou par une intelligence artificielle. Les 3 autres indiquent la même chose mais s'affichent après avoir terminé l'entretien. Cette page n'existait pas dans la précédente étude et le mécanisme qui permet d'associer une condition au candidat a été ajouté.

7. Consignes enregistrement

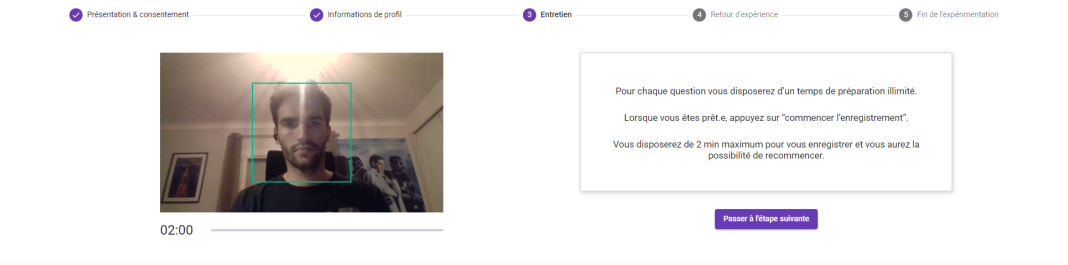


Figure : Capture d'écran de la plateforme : consignes d'enregistrement.

Cette étape permet de donner des consignes pour l'enregistrement des vidéos d'entretien. Encore une fois, on garde sur cette page la même disposition que le reste des pages de la partie entretien. Le code est donc réutilisé facilement grâce au système de component que propose Angular.

8. Questions entretien

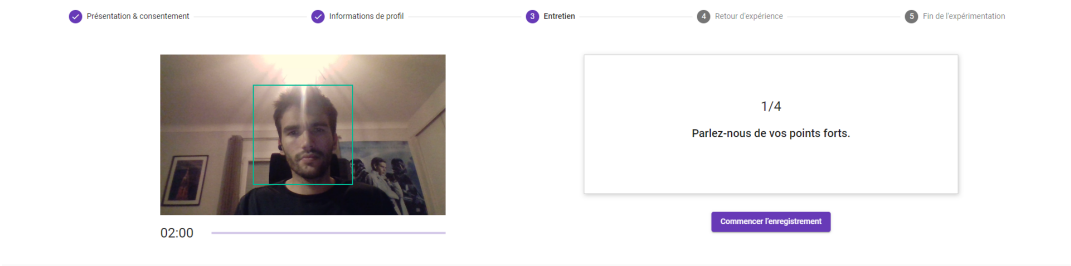


Figure : Capture d'écran de la plateforme : questions de l'entretien 1/2

Il existe 4 questions pour l'entretien. Ces 4 questions suivent le même principe. D'abord la question s'affiche et l'utilisateur peut s'entraîner à répondre. L'utilisateur peut ensuite commencer à s'enregistrer. L'enregistrement vidéo peut durer jusqu'à 2 minutes.

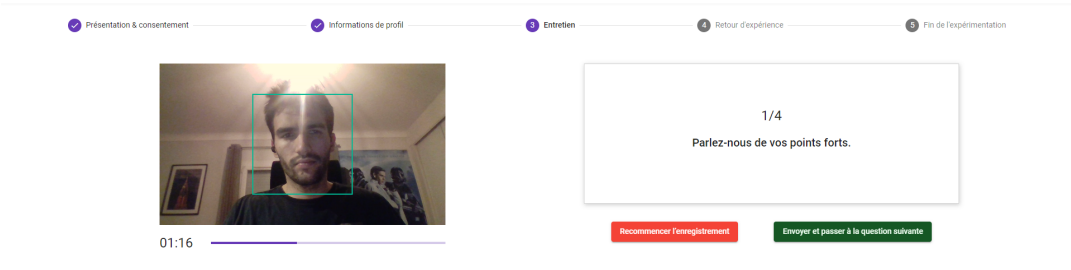


Figure : Capture d'écran de la plateforme : questions de l'entretien 2/2.

Une fois l'enregistrement terminé, l'utilisateur à la possibilité de recommencer l'enregistrement. Celui-ci peut s'enregistrer autant de fois qu'il veut, seulement le dernier enregistrement est conservé. Si l'enregistrement lui convient, il peut passer à la question suivante.

Le temps d'entraînement pour chaque question est stocké dans la base de données. Il s'incrémente dès lors que le candidat peut voir la question et s'arrête dès qu'il enregistre la vidéo finale.

9. Retour expérience

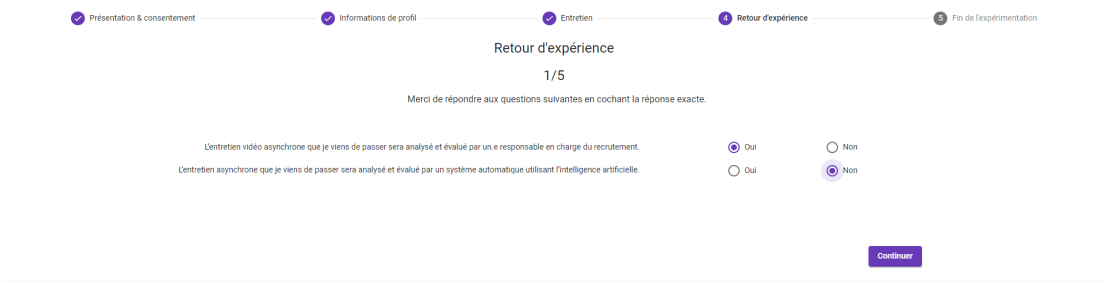


Figure : Capture d'écran de la plateforme : retour d'expérience.

Une fois l'entretien terminé, il est demandé aux utilisateurs de répondre à des questions pour avoir un retour sur leurs ressentis pendant l'expérience. Plusieurs types de questions sont possibles, répondre Oui/Non, une valeur de 1 à 7, Vrai/Faux ou une question ouverte.

10. Envoi des données

Quand toutes les réponses ont été remplies, celles-ci sont envoyées au serveur ainsi que les vidéos et le formulaire de consentement.

Voici donc le déroulement de l'étude. Celui-ci est linéaire, c'est-à-dire que l'utilisateur est obligé de faire toutes les étapes dans l'ordre. Il n'est pas possible pour lui de sauter une étape ou de revenir en arrière.

Back-end

La partie back-end était également fournie avec le projet initial. Celui-ci utilise express js pour créer des routes comme le stockage des vidéos et des fichiers PDF.

```
app.post(
  "/api/upload/video",
  multer({
    storage: multer.diskStorage({
      destination: (req, file, cb) => {
        console.log(file.fieldname);
        cb(null, "videos/");
      },
      filename: (req, file, cb) => {
        cb(null, file.originalname);
      },
    }),
  }).single("video"),
  function (req, res) {
    if (!req.file) {
      console.log("No file received");
      return res.send({
        success: false,
      });
    } else {
      console.log("file received successfully");
      return res.send({
        success: true,
      });
    }
  }
);
```

Figure : Extrait du code : stockage des fichiers vidéo.

Dans cet exemple, sur la route /api/upload/video les fichiers envoyés sont transmis dans le dossier /videos/. Une même route est faite pour l'enregistrement des fichiers PDF.

La base de données est réalisée avec SQLite, nous avons redéfinit les tables pour que celles-ci correspondent au besoin exprimé par les clients.

Nous avons donc créé 6 tables avec chacune un rôle important dans la base :

- La table DataUser contient toutes les informations relatives au profil du candidat. Celle-ci a été légèrement modifiée par rapport à l'ancienne plateforme pour prendre en compte les nouvelles informations telles que le niveau de diplôme ou le type de diplôme.
- La table DataQuestion a ajoutée afin de stocker les reponses du questionnaire de retour d'expérience pour un User. Les questions étant définies par blocs dans le protocole nous avons choisi de donner le numéro de bloc et le numéro à l'intérieur de ce bloc en tant que clé. Seule la question de fin sur l'expérience détaillée diffère du type des autres questions et est donc stockée directement dans la table User au format texte.
- La table DataVideoInterview permet de stocker pour chaque question de l'entretien des informations relatives au temps de préparation, au nombre de tentatives ainsi que le chemin d'accès à la vidéo.
- La table DataClickLogs permet de stocker les clics de l'utilisateur pendant la partie entretien. Cette table a été conservée de l'ancienne version. Précédemment, les logs du clavier et la heatmap de position du curseur étaient

également stockés. Nous les avons retirés car inutiles pour les besoins de cette enquête.

- La table DataConsentement stocke les informations liées au consentement de l'utilisateur. Son nom et son prénom sont également stockés et associés à un numéro d'anonymat (idConsent). Ceci permet de retrouver l'utilisateur et de supprimer ses données s' il le souhaite.
- La table DataLink fait un lien entre le numéro d'anonymat et l'identifiant de l'utilisateur. En supprimant cette table on supprime tout lien entre les informations récoltés et un utilisateur permettant une anonymisation complète des données.

Pour enregistrer et récupérer des données nous avons utilisé GraphQL.

Voici un exemple pour ajouter des valeurs dans la table DataAnswer.
Du côté du back-end, on définit tout d'abord le type DataAnswer qui correspond à la base ainsi que les types utilisés pour ces attributs. Puis on définit la mutation addDataAnswer en indiquant les valeurs à passer dans la requête.

```
type DataAnswer{
    idUser: Int;
    idBloc: Int;
    idQuestion: Int;
    answer: Int;
  }
```

Figure : Extrait du code : type DataAnswer.

```
type Mutation{
  addDataAnswer(idUser: Int, idBloc: Int, idQuestion: Int, answer: Int): DataAnswer
  }
```

Figure : Extrait du code : type Mutation addDataAnswer.

On ajoute ensuite la fonction qui va exécuter la requête pour ajouter les valeurs dans la base.

```
addDataAnswer: (args) =>
    query(
      `INSERT INTO DataAnswer (idUser, idBloc, idQuestion, answer) VALUES
("${args.idUser}", "${args.idBloc}", "${args.idQuestion}", "${args.answer}")`),
```

Figure : Extrait du code : Fonction ajout des données dans la table DataAnswer.

Du côté du front-end on utilise l'outil Apollo de GraphQL qui permet d'envoyer et recevoir des données en lui passant les informations nécessaires. D'abord on définit de nouveau la mutation pour indiquer les valeurs ainsi que les types nécessaires.

```
const MUTATION_DataAnswer = gql`
  mutation($idUser: Int, $idBloc: Int, $idQuestion: Int, $answer: Int) {
    addDataAnswer(idUser: $idUser, idBloc: $idBloc, idQuestion: $idQuestion, answer:
$answer) {
      idUser
      idBloc
      idQuestion
      answer
    }}
`
```

Figure : Extrait du code : Mutation DataAnswer côté FrontEnd pour indiquer les variables à passer.

Pour envoyer les données on utilise la fonction mutate de Apollo en lui passant la mutation créée précédemment. On passe également les variables qui seront stockées.

```
this.apollo
        .mutate({
          mutation: MUTATION_DataAnswer,
          variables: {
            idUser: this.numeroAnonymatDataLink,
            idBloc: parseInt(this.dataParticipant.dataAnswers.idBloc),
            idQuestion: parseInt(this.dataParticipant.dataAnswers.id),
            answer: parseInt(this.dataParticipant.dataAnswers.answer)
          },
        })
        .subscribe(
          () => {},
          (error) => {
            console.log("ERREUR DataVideo", error);
          }
        );
```

Figure : Extrait du code : Fonction d'envoi des données.

Gestion de projet

Méthodologie AGILE

La méthode agile aborde des points précis comme la flexibilité et la réactivité. Le but est de se fixer des objectifs courts et de réaliser des sprints d'une semaine pour livrer régulièrement des versions du projet et voir si le projet ne part pas dans une direction non souhaitée. Au début de chaque sprint, on a planifié les tâches que nous allions réaliser dans la semaine et la répartition du travail en fonction du nombre de tâches.

Weekly meetings

La méthode AGILE permet également d'être plus proche avec le client, car on communique régulièrement. En effet, nous avons tenu un rythme d'une réunion par semaine avec l'ensemble des porteurs de projets durant lesquels nous faisions état de notre avancée, des problèmes rencontrés, de notre planning pour les semaines suivantes ainsi que d'éventuelles questions. Ces réunions étaient également d'avoir un retour client "en temps réel" pendant le déroulement du projet ce qui nous a permis d'avancer sereinement.

Evaluation des contraintes

Au tout début de notre projet, nous avons défini précisément quelles étaient les contraintes. Les contraintes temporelles comprennent la réalisation de tous les livrables (poster, flyer, rapport), la préparation aux soutenances, les vacances, ainsi que les créneaux de cours. Les contraintes techniques et légales comprennent l'utilisation des technologies déjà existantes dans le projet ainsi que le respect du RGPD.

En prenant en compte ces contraintes dès le départ nous avons pu prévoir au mieux notre planning et certaines tâches qui leur sont associées.

Evaluation des risques

Dans l'évaluation des risques nous avons identifié une potentielle panne ou vol de notre matériel informatique. C'est pourquoi nous avons fixé une règle nous imposant de sauvegarder régulièrement et en chaque fin de journée notre travail en ligne (grâce au gestionnaire de version Git).

Trello

Afin de faciliter la gestion de notre projet, nous avons décidé d'utiliser l'application trello.

Trello est un outil de gestion de projet innovant basé sur des tableaux kanban, qui permet de visualiser facilement l'état d'avancement des tâches et des projets en temps réel. Il permet de gérer efficacement les priorités et le respect de l'avancement du planning. Chaque carte représentant une tâche sur laquelle nous pouvons ajouter un membre et définir des étiquettes (correspondant au type de tâche) celà nous a grandement facilité le partage des tâches.

Aussi, dans le cadre de la méthode AGILE, nous avons pu définir nos sprints par ecrit et sous forme de cartes que nous avions simplement à déplacer entre les listes suivant leur avancement (à faire, en cours, pour plus tard...).

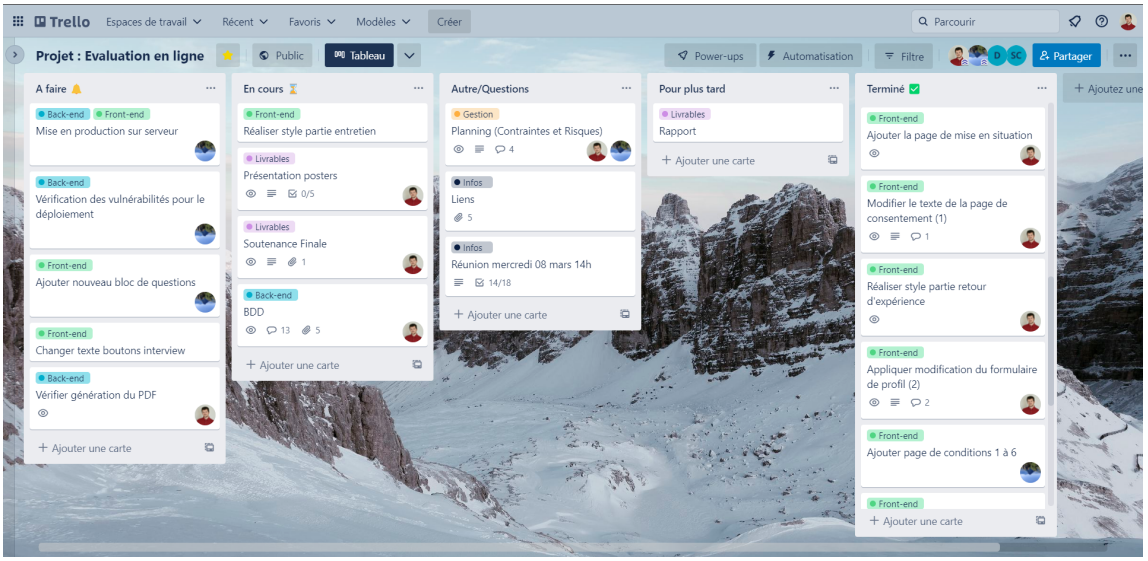


Figure : Capture d'écran de notre tableau Trello de projet.

Ayant accès à notre tableau de projet, les clients ont pu consulter en temps réel l'avancement du projet.

Rôles des membres

Bien que nous ayons travaillé le plus souvent à deux, nous pouvons tout de même nous attribuer certains rôles :

- Etienne (Chef de projet) s'est principalement penché sur la réalisation du back-end, du déploiement, du front-end et de la communication avec les clients.
- Angelo s'est lui concentré sur la conception des maquettes, la conception du modèle BDD, les livrables (poster/flyer/rapport), et a travaillé majoritairement sur la partie front-end.

Outils de collaboration

Comme nous l'avons présenté précédemment nous avons utilisé la plateforme Trello pour la gestion de projet et l'application Whimsical pour la réalisation de nos maquettes.

Nous avons utilisé l'outil Gitlab imposé par le client pour le partage du code. Comme nous devions travailler sur un projet déjà existant, une branche à été créée à cet effet. Nous l'avons par la suite séparée en deux nouvelles branches pour séparer nos modifications sur nos branches respectives et eviter les conflits.

En dehors des heures de travail à Polytech ou pendant les jours de télétravail, nous avons utilisé la plateforme Discord pour notre communication interne. Elle permet d'écrire, de discuter rapidement et de passer des appels.

Pour la réalisation des livrables (rapport/slides) nous avons utilisé la suite Google Drive qui permet notamment le travail collaboratif en temps réel sur un même document.

Métriques logiciels

Lignes de code & commits

3313 additions and 5493 deletions

- Etienne : 27 commits soit 64,3%
- Angelo : 15 commits soit 35,7%

Langages

En termes de langages, notre projet est découpé ainsi :

- TypeScript : 71%
- HTML : 17%
- CSS : 8%
- Javascript : 4%

Conclusion

Nous sommes satisfait de conclure ce projet sur une note positive. Les attentes du client ont été respectées et nous avons apprécié travailler sur ce projet.

Le serveur de LIG n'ayant pas été mis à notre disposition à temps pour permettre le déploiement final de la plateforme, nous avons su rebondir en déployant une version "betâ" sur un hébergeur personnel. Celà nous a permis de montrer notre travail aux clients, de faire tester le bon fonctionnement et d'intégrer les remarques et le retour des clients.

Ce projet a également été enrichissant aussi bien sur le plan technique que relationnel. L'entente dans le groupe a été bonne et il a été agréable de travailler dans ces conditions tout au long du projet.

Enfin, nous tenons à remercier Christelle Martin-Lacroux, Sybille Caffiau et Sophie Dupuy-Chessa pour leur accompagnement, leurs conseils et le temps qu'elles nous ont accordé pendant toute la durée du projet.

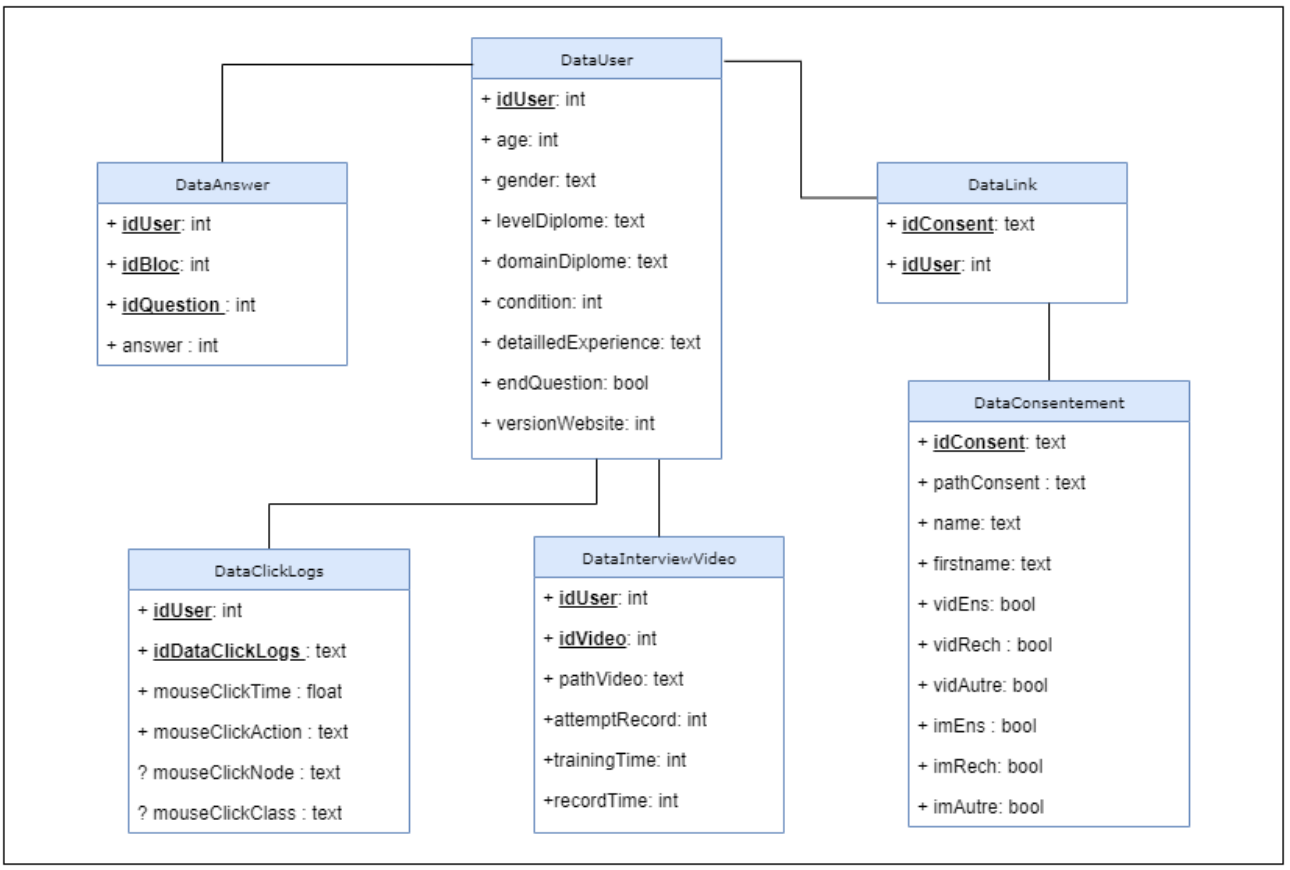
Démonstration

Une vidéo montrant une utilisation complète de la plateforme est disponible à ce lien :

https://drive.google.com/file/d/1_jo8QKOnqi9shgNRHKGxtOzAb-0dl7Ok/view?usp=sharing

Annexes

Annexe 1 : Modèle de base de données



Annexe 2 : Maquette de l'interface utilisateur

Lien externe vers notre maquette Whimsical :

https://whimsical.com/maquette-eval-en-ligne-4yKzA5CJL77UW9VtrW1j6v