R5.DWeb-DI.05 - Développement front avancé

NEXT.JS

A. Problématique :

L'établissement des emplois du temps de la formation MMI ou plus généralement de n'importe quelle formation universitaire, est quelque chose de complexe. En amont, il est nécessaire de collecter les disponibilités de tous les intervenants. Et ce n'est pas forcément simple non plus :

Sachant que l'ensemble des cours d'un BUT peut s'étirer sur 35 semaines.
Sachant qu'en moyenne, une semaine c'est 32h de cours.
Et sachant (en simplifiant) qu'un créneau pour un cours est en moyenne de 2h.
Cela signifie que chaque intervenant devrait donner sa disponibilité pour chaque créneau possible, soit 16 créneaux par semaine et donc 512 créneaux par an.
Enfin, sachant que l'ensemble des cours est assuré en moyenne par 50 intervenants, il en résulte qu'avant même d'établir les emplois du temps, il faudrait collecter 25600 disponibilités.

Evidemment, ce n'est pas réaliste et l'on peut simplifier heureusement les choses :

- Des intervenants peuvent avoir toujours les mêmes disponibilités, quelle que soit la semaine.
- Des intervenants peuvent avoir toujours les mêmes disponibilités, sauf certaines semaines qui doivent être précisées.
- Des intervenants n'ont pas besoin de donner leur disponibilité en dehors des semaines où ils ont cours.

Mais il faut aussi prévoir le cas des intervenants qui auront cours presque toutes les semaines et leurs disponibilités seront différentes chaque semaine.

B. Objectif :

A l'aide du framework Next.js, développer une application qui comportera deux parties :

1. **Accès administrateur à un dashboard :**
   L'accès au dashboard est soumis à authentification. Il faut posséder un compte administrateur et saisir un login et mot de passe pour y accéder.
   Le dashboard possède les fonctionnalités suivantes :

- Gérer la liste des intervenants.
  Pour chaque intervenant on enregistrera son nom, son prénom, son email (qui doit être unique dans la base). On enregistrera également une clé (unique) d'accès (une chaîne), la date de création de cette clé ainsi qu'une date limite de validité pour cette clé.
  Le dashboard doit permettre à l'administrateur de :
  - consulter la liste des intervenants
  - ajouter un nouvel intervenant
  - supprimer un intervenant
  - éditer les informations d'un intervenant
  - générer ou régénérer la clé d'un intervenant
  - éditer la date limite de validité de la clé

- Visualiser/consulter les disponibilités pour chaque intervenant :
  En sélectionnant un intervenant et une semaine, l'administrateur peut voir les disponibilités de l'intervenant sélectionné pour la semaine sélectionnée.

- Exporter en JSON (même format que le fichier d'exemple qui vous a été donné) les disponibilités de tous les intervenants.

- Éditer (et sauvegarder) les disponibilités d'un intervenant.

2. **Accès intervenant pour la saisie de ses disponibilités :**
   Via un lien qui lui sera communiqué, un intervenant peut accéder à une partie de l'application pour saisir ses disponibilités pour l'année universitaire à venir. Le lien comprendra la clé de l'intervenant ce qui permettra à l'application de l'identifier (on a bien dit "identifier" et pas "authentifier") sous réserve que la date limite associée ne soit pas dépassée (sinon on affichera un message de type "Votre lien a expiré, veuillez contacter l'administrateur". Alternativement on pourra aussi choisir de permettre quand même l'accès aux disponibilités déjà saisies, mais en lecture seule).

   Vous serez très attentifs aux points suivants :

   - La saisie des disponibilitées doit être la plus simple et directe possible

   - L'intervenant sélectionne une semaine pour donner ses disponibilités pour la semaine sélectionnée. Via une interface de type "calendrier de la semaine", l'utilisateur place des blocs qui matérialisent les créneaux où il est disponible.

   - Pour une semaine donnée, l'intervenant doit pouvoir définir qu'il s'agit de ses disponibilités par défaut (i.e disponibilités à appliquer pour toutes les semaines pour lesquelles il n'aura pas saisi de disponibilité spécifique). Note : L'intervenant

n’est pas non plus obligé de donner des disponibilités par défaut.

- Pour une semaine donnée, l’intervenant doit pouvoir rapidement sélectionner les autres semaines pour lesquelles les mêmes disponibilités sont à appliquer.

- L’enregistrement (en base) des disponibilités se fait automatiquement à chaque modification (pas de bouton “Enregistrer”).

- Il est possible de faire la saisie des disponibilités en plusieurs fois tant que le lien n’a pas expiré.

Il n’est pas demandé que l’application gère elle-même l’envoi de chaque lien d’accès à chaque intervenant (éventuellement à terme selon le temps restant).

Recommandations graphique/ergonomique :
Les interfaces de l’application doivent être neutres et “épurées” au maximum afin de favoriser la lisibilité et de mettre en évidence les fonctionnalités disponibles. Vous êtes fortement incités à utiliser des bibliothèques de composants “clés en main” afin d’assurer la cohérence de vos interfaces et d’optimiser le temps de développement car les délais sont serrés. Tailwind/shadcn est un exemple parmi d’autres.

C. Itérations

Comme d’habitude, vous procéderez par itérations, chaque itération correspondant plus ou moins à une fonctionnalité de l’application. L’ordre dans lequel procéder n’est pas forcément unique. Mais dans tous les cas, il ne faut pas suivre l’ordre dans lequel les fonctionnalités ont été présentées dans la précédente partie !

Pour vous aider, vous pouvez vous appuyer sur la proposition suivante :

Itération 0 : Mise en place de l’environnement de développement

L’environnement de développement fera l’objet d’un (multi)conteneur Docker. Vous aurez besoin d’un environnement Node.js pour faire tourner un projet Next.js / Tailwind / Typescript. Vous aurez besoin d’un serveur de base de données relationnelles (MySQL, PostgreSQL, MariaDB…). Accessoirement, vous pouvez prévoir un outil d’administration de base de données. Enfin votre projet fera l’objet d’un repo Github et doit pouvoir tourner dans un codespace (ce qui doit être le cas sans rien faire de plus si votre Docker est bien fonctionnel).

Itération 1 : Configuration du projet Next.js

La configuration Tailwind / Typescript est assurée, normalement, à l’initialisation du votre projet Next.js. Vous devez toutefois configurer votre environnement afin de permettre à votre

application de se connecter à votre base de données. Attention, c'est globalement similaire à ce que vous avez fait dans le tuto Next.js, sauf qu'ici, la base n'est pas hébergée sur Vercel, mais se trouve dans votre Docker.

Itération 2 : Création du dashboard (vide) d'administration
Créer des routes/pages pour votre dashboard d'administration. Prévoyez un menu avec 2 entrées. Une entrée "intervenants" qui permettra d'accéder à la gestion des intervenants. Une entrée "disponibilités" qui permettra d'accéder à la consultation des disponibilités des intervenants. Dans l'immédiat, cliquer sur les menus affichera uniquement une page vide avec juste un message de type "Gestion des Intervenants" ou "Disponibilités des intervenants".
Notes : Cette itération est analogue pour ne pas dire identique à certaines "leçons" du tuto Next.js. Basez-vous dessus. Respectez bien la logique du framework et l'architecture du projet en complétant votre bibliothèque de composants dans le dossier "ui" et en important ces composants sur les pages ou layout où ils sont utiles.

—-
Notes : les itérations 3 à 10 qui suivent sont également analogues à la gestion des "customers" réalisées lors du tuto Next.js. Tout comme la partie authentification. Basez vous dessus, c'est quasi la même chose. Ça doit aller très vite.
—-

Itération 3 : [BDD] Ajouts des intervenants
Dans votre base de données, créer une table Intervenants en cohérence avec la description des informations à stocker dans la partie B :
Intervenants(id, email, firstname, lastname, key, creationdate, enddate, availability)
Enregistrez quelques intervenants "anonymes" (par exemple "intervenant.A@unilim.fr", "Intervenant", "A", "A-key", "10/11/2024", "11/05/2025", ""). Dans l'immédiat vous pouvez mettre ce que vous voulez pour la clé et les dates.
Note : la colonne availability servira plus tard à stocker les disponibilités d'un intervenant sous la forme d'une chaîne JSON. Selon la base que vous utilisez, un type de données JSON peut exister. A défaut, prévoyez de stocker une grande chaîne de caractères. Cette approche peut paraître assez "violente" mais nous expliquerons en temps utile pourquoi elle se justifie ici.

Itération 4 : Création de la page Gestion des intervenants - lecture
Complétez la page "Gestion des intervenants" de sorte à afficher la liste des intervenants présents en base de données. On signalera (couleur, icône… comme vous voulez) les intervenants dont la date de validité de la clé est dépassée.

Itération 5 : Création de la page Gestion des intervenants - suppression
Permettre la suppression d'un intervenant via un icône (une croix, une poubelle…).

Itération 6 : Création de la page Gestion des intervenants - ajout
Prévoir un bouton "add" pour permettre d'ajouter un nouvel intervenant dans la base via un formulaire. Préalablement à l'enregistrement, pensez à vérifier la complétude du formulaire et

pensez à traiter les cas d'erreur, par exemple si l'email existe déjà en base (si vous avez bien défini dans votre base que l'email est unique, l'insertion d'une nouvelle ligne avec un email existant échouera).
Votre formulaire ne permettra pas de saisir la clé et ses dates de création et validité. Ces informations seront établies automatiquement avant l'enregistrement en base. Pour ce, générez un identifiant unique en guise de clé. Plusieurs solutions sont possibles. La plus fiable est peut-être d'utiliser une librairie spécifique telle que uuid que vous pouvez installer sur votre projet via npm. La date de création de la clé sera bien entendu celle du jour de sa création et par défaut, sa date limite de validité sera 2 mois plus tard.

Itération 7 : Création de la page Gestion des intervenants - édition
Prévoir pour chaque intervenant la possibilité de modifier les informations présentes à l'exception de l'identifiant et de la clé et de sa date de création (mais on doit pouvoir modifier la date de validité)

Itération 8 : Création de la page Gestion des intervenants - régénérer la clé
Prévoir la possibilité pour chaque intervenant de générer une nouvelle clé via un icône, ainsi que la possibilité de régénérer les clés de tous les intervenants.
Pourquoi faire peut-on se demander ? Le mécanisme d'identification par clé reste limité. Celui qui accède est celui qui possède la clé. Par sécurité, l'administrateur peut souhaiter vouloir régénérer la clé d'un intervenant voire de tous les intervenants. Les dates de création et de validité sont alors mise à jour comme lors de l'ajout d'un nouvel intervenant.

Itération 9 : [BDD] Ajout des administrateurs
Ajouter une table Users pour enregistrer un ou plusieurs comptes administrateur de l'application. Le login sera une adresse mail. Insérer au moins un compte dans votre table. Là encore, basez-vous sur le tuto Next.js.

Itération 10 : Protection de l'accèd au dashboard d'administration
Modifier votre application pour l'accès au dashboard soit limité aux administrateurs authentifiés. Dans le cas contraire l'utilisateur est invité à compléter un formulaire login/password.