Université de Lorraine – IDMC – L3 MIAGE

Rapport CSI Étape 3 – LIVRE D'AILLEURS

Modélisation des fonctionnalités de notre application, modèle logique et physique, implantation des contraintes d'intégrités et description de l'application web

SEBTI Célestin, WALTER Hugo, BALLOIR Gael, HUBERT Antoine, CIAPA Tom
04/04/2024

Table des matières

Introduction 4
Règles de gestion 5
Acteurs 6
Modèle de données (diagramme conceptuel UML) 7
Modèle logique 8
Modèle physique en intention 9
Dictionnaire des données 10
Contraintes d'intégrité statiques et dynamiques 16
Fonctionnement d'une édition d'un festival (intéractions & déroulement) 23
Description de l'implantation (application web) 24
- Structure du projet 24
- Stack : description des frameworks 24
- Installation de notre application 26
- Notre application 27
  - Dashboard auteur 28
  - Dashboard établissement 31
  - Dashboard commission scolaire 34
- Fonctionnement technique 39
  - Routes pour l'authentification 39
  - Route pour proposer un ouvrage 43
  - Route pour récuperer les interventions d'un établissement dans une édition donnée 45
  - Un composant React en détail 47

Modélisation des fonctionnalités 51
- Inscription d'un établissement ou d'un auteur 51
- Connexion d'un établissement ou d'un auteur 51
- Consulter les ouvrages 52
- Consulter les établissements 52
- Consulter les auteurs 52
- Ajouter un ouvrage 53
- Rechercher un ouvrage 53

Modifier un ouvrage 53
Supprimer un ouvrage 54
Rechercher un auteur 54
Modifier un auteur 54
Supprimer un auteur 55
Rechercher un établissement 55
Modifier un établissement 56
Supprimer un établissement 56
Consultation des interventions 56
Consultation des interventions d'un établissement spécifique 57
Consultation des voeux 57
Consultation des voeux d'un établissement spécifique 57
Notifier sa volonté de participer 58
Générer les statistiques sur les ouvrages 58
Consulter les statistiques sur les ouvrages 58
Générer les statistiques sur la campagne de voeux 59
Consulter les statistiques sur la campagne de voeux 59
Générer les statistiques sur les interventions 59
Consulter les statistiques sur les interventions 60
Création d'une édition 60
Modification d'une édition 60
Sélection d'ouvrages 61
Sélection d'établissements 61
Fin des inscriptions : on prévient les auteurs et établissements participants 61
Fin de la campagne de voeux : récapitulatif des interventions à venir et refus automatique des voeux 62
Fin de l'édition (Automatique) : 62
Enregistrer un accompagnateur ou un interprète 62
Émettre un Vœu 63
Modifier un vœu 63
Supprimer un vœu : 64
Planifier une intervention 64

Annuler une intervention 65
Remplacer une intervention..... 65
Renseigner le nombre d'élèves..... 66
Repartition du travail au sein du groupe 67
Conclusion 68

Introduction

L'objectif du projet est de réaliser un système d'information sous forme d'une application web permettant l'organisation et la gestion du festival littéraire international "Livres D'ailleurs".

Ce système d'information a pour objectif la gestion et le stockage des différentes données nécessaires à la planification automatique des interventions des auteurs participants dans différents établissements pour chaque édition du festival.

Il devra présenter une interface web permettant l'inscription et l'authentification des auteurs et établissements concernés afin qu'ils puissent réaliser facilement différentes actions tout au long du processus (se référer aux fonctionnalités énumérées plus bas).

Il devra, pour chaque édition, mettre en place une campagne de vœux afin de permettre aux établissements de choisir parmi une sélection d'ouvrages, portant sur un thème donné, selon un principe de priorité des vœux, et de fournir pour chaque vœu, les coordonnées d'un référent d'établissement.

Il devra être capable, après une campagne de vœux, de planifier algorithmiquement les interventions des auteurs selon différentes contraintes dont la priorité des vœux et la localisation des établissements. Pour chaque intervention devront être fournies les coordonnées d'un accompagnateur et si besoin d'un interprète.

Il devra, à la fin de chaque édition, générer des statistiques sur l'édition qui a eu lieu ainsi que garder en mémoire les informations sur les auteurs, établissements, référents, accompagnateurs et interprètes pour des prochaines éditions.

Dans cet intérêt ce rapport présentera la troisième étape de conception du système d'information. Seront présentés les différents acteurs du système prévus, une retranscription de la demande via la liste des différentes règles de gestion prévues, les différents diagramme de classe, l'implantation de notre SI sous forme de site web et la modélisation des différentes fonctionnalités qui en découlent.

Règles de gestion

- R0 : Le système doit être réactualisé avant chaque édition du festival pour prendre en compte les nouvelles informations sur les auteurs, les ouvrages, les interventions, les accompagnateurs, les interprètes, et les référents
- R1 : Le système doit conserver des informations sur les auteurs et les établissements, qu'ils aient participé récemment ou non au festival.
- R2 : Le système doit stocker les informations suivantes pour chaque auteur : nom, coordonnées (adresse, numéro de téléphone, adresse e-mail), et les langues qu'il maîtrise à l'oral et à l'écrit.
- R3 : Chaque auteur participant au festival doit proposer au moins un ouvrage pour être considéré
- R4 : Le système doit enregistrer les informations suivantes pour chaque ouvrage : nom, classes concernées, public ciblé.
- R5 : Le système doit lancer une campagne de vœux pour une période définie avant chaque édition du festival.
- R6 : Le système doit stocker les informations suivantes pour chaque établissement : nom, coordonnées, localisation, type (université, lycée général, lycée professionnel, collège, école primaire ou maternelle, établissement médico-sociaux, établissement pénitentiaire), et un référent (nom, coordonnées)
- R7 : Chaque établissement a le droit de formuler jusqu'à trois vœux, en indiquant leur ordre de priorité.
- R8 : À la fin de la campagne de vœux, le système doit planifier pour chaque établissement la liste des interventions d'auteurs, en respectant les préférences exprimées par les établissements.
- R9 : Chaque intervention doit être accompagnée d'un accompagnateur, dont les informations (nom, coordonnées) sont enregistrées.
- R10 : Selon les langues maîtrisées par l'auteur et la demande de l'établissement, chaque intervention peut être accompagnée d'un interprète, dont les informations (nom, coordonnées) sont enregistrées.
- R11 : Chaque intervention d'auteur doit avoir une durée comprise entre 2 et 4 heures.
- R12 : Chaque auteur peut effectuer jusqu'à 3 interventions par jour.
- R13 : Le système doit optimiser la planification des interventions en tenant compte de la localisation géographique des établissements pour minimiser les déplacements.
- R14 : En cas de désistement d'un auteur ou d'un établissement, le système doit permettre la possibilité de remplacer l'intervention annulée.
- R15 : Le système doit générer des statistiques, y compris le nombre d'ouvrages proposés lors de chaque édition du festival, le taux de participation des

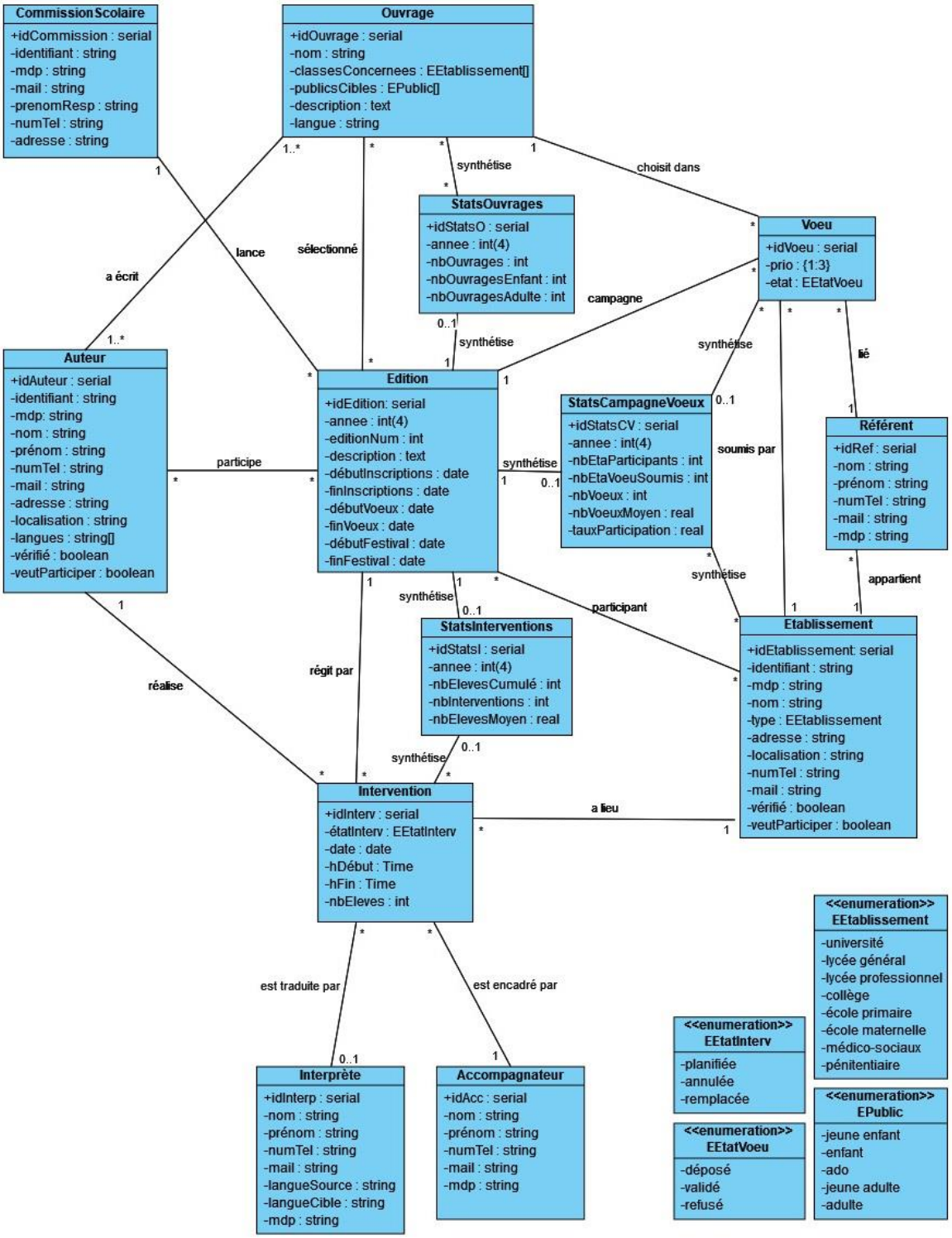
établissements à la campagne de vœux par édition, et le nombre d'élèves présents lors de chaque intervention.

- R16 : Le système doit prendre en compte les disponibilités des auteurs pour planifier les interventions en fonction de leurs agendas.
- R17 : Les établissements doivent être informés des interventions planifiées, y compris la date, l'heure, et les détails sur l'auteur et l'ouvrage, suffisamment à l'avance pour organiser leur programme.
- R18 : En cas de changement dans les informations des acteurs, des établissements, des référents, des accompagnateurs, des interprètes ou des ouvrages le système doit permettre la mise à jour de ces données.
- R19 : Le système doit maintenir un historique des éditions précédentes du festival, y compris les données sur les auteurs, les établissements, et les statistiques, pour référence future.

Acteurs

Nous avons identifé les acteurs suivants : Gestionnaire du système, Commission Scolaire, Auteur, Etablissement, Interprète, Accompagnateur, Référent. Sachant que Accompagnateur et Interprète sont des acteurs plus secondaires, tout comme référent.

Modèle de données (diagramme conceptuel UML)



Modèle logique

Modèle physique en intention

Nous avons rajouté deux domain en plus des types EEtablissement, Epublic, EEtatVoeu, EEtatInterv :

CREATE DOMAIN EmailAddress VARCHAR(255) CHECK (VALUE ~* '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$') NOT NULL;

CREATE DOMAIN MotDePasse VARCHAR(255) CHECK (LENGTH(VALUE) >= 8) NOT NULL;

On utilise ces deux domaines directement dans les tables pour les attributs ‘mdp’ et ‘mail’.

Voici le modèle en intention :

CommissionScolaire(idCommission : serial, identifiant : string, mdp : MotDePasse, mail : EmailAddress, prenomResp : string, nomResp : string, numTel : string , adresse : string)

Ouvrage(idOuvrage : serial, nom : string, classesConcernees : EEtablissement[], publicsCibles : EPublic[], description : text, langue : string)

Ouvrage_Auteur(#idOuvrage : serial, #idAuteur : serial)

Auteur(idAuteur : serial, identifiant : string, mdp : MotDePasse, nom : string, prénom : string, numTel : string, mail : EmailAddress, adresse : string, localisation : string, langues : string[], verifié : boolean, veutParticiper : boolean)

Edition(idEdition : serial, editionNum : int, année : int, description : text, debutInscriptions : date, finInscriptions : date, débutVoeux : date, finVoeux : date, débutFestival : fate, finFestival : date, #idStatsInterv : int, #idStatsCampagneVoeux : int, #idStatsOuvrages : int)

Auteur_Edition(#idAuteur : int, #idEdition : int)

Ouvrage_Edition(#idOuvrage : int, #idEdition : int)

Intervention(idInterv : serial, étatInterv : EEtatInterv, date : date, hDébut : time, hFin : time, nbElevbes : int, #idAuteur : int, #idInterp, #idAcc :int, #idEdtion : int, #idEtablissement : int, #idStatsI : int)

Interprete(idInterp : serial, nom : string, prenom : string, numTel : int, mail : EmailAddress, mdp : MotDePasse, langueSource : string, langueCible : string)

Accompagnateur(idAcc : serial, nom : string, prenom : string, numTel : int, mail : EmailAddress, mdp : MotDePasse)

Etablissement(idEtablissement : serial, identifiant : string, mdp : MotDePasse, nom : string, type :EEtablissement, adresse : string, localisation : string, numTel : int, mail : EmailAddress, verifié : boolean, veutParticiper : boolean)

Etablissement_Edition(#idEtablissement : int, #idEdition : int)

Référent(idRef : int, nom : string, prenom : string, numTel : int, mail : EmailAddress, mdp : MotDePasse, #idEtablissement : int)

Vœu(idVoeu : serial, prio : int, etat : EEtatVoeu, #idEtablissement : string, #idOuvrage : int, #idRef : int, #idEdition : int, #idStatsCV : int)

StatsInterventions(idStatsI : serial, annee : int, nbElevesCumulé : int, nbInterventions : int, nbElevesMoyen : real)

StatsCampagneVoeux(idStatsCV : serial, annee : int, nbEtablissements : int, nbEtaParticipants : int, nbEtaVoeuSoumis : int, nbVoeux : int, nbVoeuxMoyen : real, tauxParticipation : real)

StatsCampagneVoeux_Etablissement(#idStatsCampagneVoeux : int, #idEtablissement : int)

StatsOuvrages_Ouvrage(#idOuvrage : int, #idStatsOuvrages : int)

StatsOuvrages(idStatsO : serial, annee : int, nbOuvrages : int, nbAuteur : int, nbOuvragesEnfant : int, nbOuvragesAdulte : int)

EEtatInterv(planifiée, annulée, remplacée)

EEtatVoeu(déposé, validé, refusé)

EEtablissement(université, lycée general, lycée profesionnel, collège, école primaire, école maternelle, médico-sociaux, pénitentiaire)

EPublic(jeune enfant, enfant, ado, jeune adulte, adulte)

Dictionnaire des données

Nom	Désignation	Type	Entité	Remarque
idVoeu	Identifiant du vœu	serial	Voeu	Clé primaire
prio	Priorité du voeu	Int	Voeu	{1;2;3}
etat	Etat du vœu	EEtatVoeu	Voeu	

Nom	Désignation	Type	Entité	Remarque
idRef	Identifiant du référent	serial	Referent	Clé primaire
nom	Nom du référent	String	Referent	
prenom	Prénom du référent	String	Referent	
numTel	Numéro de téléphone du Referent	String	Referent	format à 10 chiffres OU '+33..'
mail	Adresse mail du référent	EmailAddress	Referent	Domain
Mdp	Mot de passe du referent	MotDePasse	Refernet	Domain

Nom	Désignation	Type	Entité	Remarque
idOuvrage	Identifiant de l'ouvrage	serial	Ouvrage	Clé primaire
nom	Nom de l'ouvrage	String	Ouvrage	
classesConcernees	Classes concernées par l'ouvrage	EEtablissement[]	Ouvrage	Tableau de type de l'enum EEtablissement
publicCible	Public ciblé par l'ouvrage	EPublic[]	Ouvrage	Tableau de type de l'enum EPublic
description	Description de l'ouvrage	String	Ouvrage	Null
langue	Langue dans lequel est écrit l'ouvrage	String	Ouvrage	

Nom	Désignation	Type	Entité	Remarque
idEtablissement	Id de l'Etablissement	serial	Etablissement	Clé primaire
identifiant	Identifiant de connexion l'Etablissement	String	Etablissement	Unique
mdp	Mot de passe de connexion de l'Etablissement	MotDePasse	Etablissement	Domain
nom	Nom de l'Etablissement	String	Etablissement	
type	Type d'Etablissement	EEtablissement	Etablissement	
adresse	Adresse de l'Etablissement	String	Etablissement	
localisation	Localisation de l'Etablissement	String	Etablissement	

numTel	Numéro de téléphone de l'Etablissement	String	Etablissement	format à 10 chiffres OU '+33..'
mail	Courriel de l'Etablissement	EmailAddress	Etablissement	Domain
verifié	L'Etablissement est il vérifié ?	boolean	Etablissement	1 = vérifié, 0 = non
veutParticiper	L'Etablissement veut il participer ?	boolean	Etablissement	1 = veut participer, 0 = non

Nom	**Désignation**	**Type**	**Entité**	**Remarque**
idEdition	Id de l'Edition	serial	Edition	Clé primaire
annee	Annee de l'Edition	Int	Edition	Taille = 4, Unique
editionNum	Numéro de l'Edition	Int	Edition	positif
description	Description	String	Edition	
debutInscriptions	Date de début des inscriptions	Date	Edition	
finInscriptions	Date limite des inscriptions	Date	Edition	> debutInscriptions
debutVoeux	Date de début d'inscription des voeux	Date	Edition	
finVoeux	Date de fin d'inscription des voeux	Date	Edition	> debutVoeux
debutFestival	Date de début du festival	Date	Edition	
FinFestival	Date de fin du festival	Date	Edition	> debutFestival

Nom	**Désignation**	**Type**	**Entité**	**Remarque**
idAuteur	Identifiant de l'auteur	serial	Auteur	Clé primaire
Identifiant	Identifiant de l'auteur	String	Auteur	Unique
Mdp	Mot de passe de l'auteur	MotDePasse	Auteur	Domain
Nom	Nom de l'auteur	String	Auteur	
Prénom	Prénom de l'auteur	String	Auteur	
NumTel	Numéro de téléphone de l'auteur	String	Auteur	format à 10 chiffres OU '+33..'
Mail	Adresse email de l'auteur	EmailAddress	Auteur	Domain

adresse	Adresse de l'auteur	String	Auteur	
localisation	Localisation de l'auteur	String	Auteur	
langues	Langues parlées par l'auteur	String[]	Auteur	
Vérifié	Dit si l'auteur est vérifié ou non	Boolean	Auteur	
veutParticiper	Dit si l'auteur veut participer ou non	Boolean	Auteur	

Nom	**Désignation**	**Type**	**Entité**	**Remarque**
idInterp	Identifiant de l'interprète	serial	Interprete	Clé primaire
Nom	Nom de l'interprète	String	Interprete	
Prénom	Prénom de l'interprète	String	Interprete	
numTel	Numéro de téléphone de l'interprète	String	Interprete	format à 10 chiffres OU '+33..'
Mail	Adresse email de l'auteur	EmailAddress	Interprete	Domain
Mdp	Mot de passe	MotDePasse	Interprete	Domain
langueSource	Langue source de l'interprète	String	Interprete	différente de langueCible
langueCible	Langue cible de l'interprète	String	Interprete	différente de langueSource

Nom	**Désignation**	**Type**	**Entité**	**Remarque**
idInterv	Identifiant de l'intervention	serial	Intervention	Clé primaire
étatInterv	État de l'intervention	EEtatInterv	Intervention	{planifiée, annulée, remplacée}
Date	Date de l'intervention	Date	Intervention	
hDébut	Heure de début d'intervention	Time	Intervention	avant hFin
hFin	Heure de fin d'intervention	Time	Intervention	après hDébut
nbEleves	Nombre d'élèves présents	Int	Intervention	Null; positif

Nom	Désignation	Type	Entité	Remarque
idAcc	Identification de l'accompagnateur	serial	Accompagnateur	Clé primaire
Nom	Nom de l'accompagnateur	String	Accompagnateur	
Prénom	Prénom de l'accompagnateur	String	Accompagnateur	
numTel	Numéro de téléphone	String	Accompagnateur	format à 10 chiffres OU '+33..'
Mail	Email de l'accompagnateur	Domain	Accompagnateur	Domain
Mdp	Mot de passe de l'accompagnateur	MotDePassse	Accompagnateur	Domain

Nom	Désignation	Type	Entité	Remarque
idStatsO	Identification de la stat	serial	StatsOuvrages	Clé primaire
annee	Année de l'édition correspondant à la statistique	Int	StatsOuvrages	Taille = 4; Unique
nbOuvrages	Nombre d'ouvrages de l'édition	Int	StatsOuvrages	positif
nbOuvragesEnfant	Nombre d'ouvrages de l'édition destinés aux enfants	Int	StatsOuvrages	positif
nbOuvragesAdulte	Nombre d'ouvrages de l'édition destinés aux adultes	Int	StatsOuvrages	positif

Nom	Désignation	Type	Entité	Remarque
idStatsCV	Identification de la statistique campagne de voeux	serial	StatsCampagneVoeux	Clé primaire
annee	Année de l'édition correspondant à la statistique	Int	StatsCampagneVoeux	Taille = 4; Uniquel
nbEtaParticipants	Nombre d'Établissements de l'édition	Int	StatsCampagneVoeux	positif
nbEtaVoeuSoumis	Nombre d'Etablissements ayant soumis au moins un	Int	StatsCampagneVoeux	positif

	vœu pour l'édition en cours			
nbVoeux	Nombre de vœux totaux	Int	StatsCampagneVoeux	positif
nbVoeuxMoyen	Nombre de vœux moyens	Numeric	StatsCampagneVoeux	nbVoeux/nbEtaParticipants
tauxParticipation	Taux de participation des Etablissements	Numeric	StatsCampagneVoeux	entre 0 et 1

Nom	**Désignation**	**Type**	**Entité**	**Remarque**
idStatsI	Identification de la statistique intervention	serial	StatsInterventions	Clé primaire
annee	Année de l'édition correspondant à la statistique	Int	StatsInterventions	Taille = 4; Unique
nbElevesCumulé	Nombre total d'élèves pour l'année	Int	StatsInterventions	positif
nbInterventions	Nombre total d'Interventions pour l'année	Int	StatsInterventions	positif
nbElevesMoyen	Nombre d'élèves moyen par Intervention pour l'année	Numeric	StatsInterventions	= nbElevesCumulé / nbInterventions

Contraintes d’intégrité statiques et dynamiques

Pour la classe **Auteur** :

- Un auteur ne peut pas changer sa volonté de participation (Auteur.veutParticiper) après la fin des inscriptions : Dynamique Forte

```
-- Un auteur ne peut pas changer sa volonté de participation (Auteur.veutParticiper) après la fin des inscriptions : Dynamique Forte

CREATE OR REPLACE FUNCTION tgVeutParticiperAuteur() RETURNS TRIGGER AS $$
DECLARE
    v_finInscriptions DATE;
BEGIN
    -- Récupération de la date de fin des inscriptions de l'édition de l'année en cours
    SELECT finInscriptions INTO v_finInscriptions FROM Edition e
        WHERE  e.annee = date_part('year', CURRENT_DATE);
    -- On lève des exceptions si la contrainte n'est pas respectée
    IF v_finInscriptions > CURRENT_DATE AND NEW.veutParticiper <> OLD.veutParticiper THEN
        RAISE EXCEPTION 'La période des iscriptions est déjà passée, l''auteur ne peut pas modifier sa volontée de participation pour l''édition';
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

-- Création du Trigger associé à la procédure stockée tgVeutParticiperAuteur
CREATE OR REPLACE TRIGGER tgVeutParticiperAuteur
    BEFORE INSERT OR UPDATE ON Auteur
    FOR EACH ROW EXECUTE PROCEDURE tgVeutParticiperAuteur();
```

- Un auteur qui participe à l’édition est necessairement verifié : Statique Forte

```
-- Un auteur qui participe à l’édition est nécessairement vérifié et veutParticiper : Statique Forte


CREATE OR REPLACE FUNCTION tgParticipationAuteur() RETURNS TRIGGER AS $$
DECLARE
    v_verifie BOOLEAN;
    v_veutParticiper BOOLEAN;
BEGIN
    -- Récupération des attributs 'verifie' et 'veutParticiper' de l'Auteur concerné
    SELECT verifie, veutParticiper INTO v_verifie, v_veutParticiper FROM Auteur a
        WHERE  NEW.idAuteur = a.idAuteur;
    -- On lève des exceptions si la contrainte n'est pas respectée
    IF v_verifie <> TRUE THEN
        RAISE EXCEPTION 'L''auteur n''est pas encore vérifié';
    ELSEIF v_veutParticiper <> TRUE THEN
        RAISE EXCEPTION 'L''auteur ne veut pas participer à l''édition';
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;


-- Création du Trigger associé à la procédure stockée tgParticipationAuteur (seuleument à l'insertiion)
CREATE OR REPLACE TRIGGER tgParticipationAuteur
    BEFORE INSERT ON Auteur_Edition
    FOR EACH ROW EXECUTE PROCEDURE tgParticipationAuteur();
```

Pour la classe **Etablissement** :

- Un etablissement ne peut pas change sa volonté de participation (Etablissement.veutParticiper) après la fin des inscriptions : Dynamique Forte

```sql
-- Un établissement ne peut pas changer sa volonté de participation (Etablissement.veutParticiper) après la fin des inscriptions : Dynamique Forte

CREATE OR REPLACE FUNCTION tgVeutParticiperEtab() RETURNS TRIGGER AS $$
DECLARE
    v_finInscriptions DATE;
BEGIN
    -- Récupération de la date de fin des inscriptions de l'édition de l'année en cours
    SELECT finInscriptions INTO v_finInscriptions FROM Edition e
        WHERE  e.annee = date_part('year', CURRENT_DATE);
    -- On lève des exceptions si la contrainte n'est pas respectée
    IF v_finInscriptions > CURRENT_DATE AND NEW.veutParticiper <> OLD.veutParticiper THEN
        RAISE EXCEPTION 'La période des iscriptions est déjà passée, l''établissement ne peut pas modifier sa volontée de participation pour l''édition';
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

-- Création du Trigger associé à la procédure stockée tgVeutParticiperEtab
CREATE OR REPLACE TRIGGER tgVeutParticiperEtab
    BEFORE INSERT OR UPDATE ON Etablissement
    FOR EACH ROW EXECUTE PROCEDURE tgVeutParticiperEtab();      You, 12 hours ago • Finished project, final commit, fixed few minor…
```

- Un établissement qui participe à l'édition est nécessairement vérifié : Statique Forte

```sql
-- Un établissement qui participe à l'édition est nécessairement vérifié et veutParticiper : Statique Forte


CREATE OR REPLACE FUNCTION tgParticipationEtab() RETURNS TRIGGER AS $$
DECLARE
    v_verifie BOOLEAN;
    v_veutParticiper BOOLEAN;
BEGIN
    -- Récupération des attributs 'verifie' et 'veutParticiper' de l'Etablissement concerné
    SELECT verifie, veutParticiper INTO v_verifie, v_veutParticiper FROM Etablissement e
        WHERE  NEW.idAuteur = e.idAuteur;
    -- On lève des exceptions si la contrainte n'est pas respectée
    IF v_verifie <> TRUE THEN
        RAISE EXCEPTION 'L''établissement n''est pas encore vérifié';
    ELSEIF v_veutParticiper <> TRUE THEN
        RAISE EXCEPTION 'L''établissement ne veut pas participer à l''édition';
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

-- Création du Trigger associé à la procédure stockée tgParticipationEtab (seuleument à l'insertiion)
CREATE OR REPLACE TRIGGER tgParticipationEtab
    BEFORE INSERT ON Etablissement_Edition
    FOR EACH ROW EXECUTE PROCEDURE tgParticipationEtab();
```

Pour la classe **Edition** :

- Edition.débutInscriptions est antérieur à Edition.finInscriptions lui même antérieur à Edition.débutVoeux lui même antérieur à Edition.finVoeux lui même antérieur à Edition.débutFestival lui même antérieur à Edition.finFestival : Statique Forte

La contrainte est directement implantée sur la table :

```
CREATE TABLE Edition(
    idEdition SERIAL,
    annee INTEGER NOT NULL UNIQUE CHECK (annee > 2000 AND annee < 2100),
    description TEXT NULL,
    debutInscritpions DATE NOT NULL,
    finInscriptions DATE NOT NULL,
    debutVoeux DATE NOT NULL,
    finVoeux DATE NOT NULL,
    debutFestival DATE NOT NULL,
    finFestival DATE NOT NULL,
    idStatsI INTEGER REFERENCES StatsInterventions,
    idStatsCV INTEGER REFERENCES StatsCampagneVoeux,
    idStatsO INTEGER REFERENCES StatsOuvrages,
    idCommission INTEGER REFERENCES CommissionScolaire,
    PRIMARY KEY (idEdition),
    CHECK (debutInscritpions < finInscriptions),
    CHECK (finInscriptions < debutVoeux),
    CHECK (debutVoeux < finVoeux),
    CHECK (finVoeux < debutFestival),
    CHECK (debutFestival < finFestival)
);
```

Pour la classe **Voeu** :

- Un voeux sont necessairement soumis entre les dates Edition.debutVoeux et Edition.finVoeux : Statique Forte

```
-- Les voeux sont obligatoirement soumis entre les dates Edition.debutVoeux et Edition.finVoeux : Statique Forte

CREATE OR REPLACE FUNCTION tgSoumissionVoeu() RETURNS TRIGGER AS $$
DECLARE
    v_finInscriptions DATE;
    v_debutInscriptions DATE;
BEGIN
    -- Récupération des dates des inscriptions de l'édition de l'année en cours
    SELECT finInscriptions, debutInscriptions INTO v_finInscriptions, v_debutInscriptions FROM Edition e
        WHERE  e.annee = date_part('year', CURRENT_DATE);
    -- On lève des exceptions si la contrainte n'est pas respectée
    IF CURRENT_DATE < v_debutInscriptions THEN
        RAISE EXCEPTION 'La période des inscriptions n''a pas encore lieu';
    ELSEIF CURRENT_DATE > v_finInscriptions THEN
        RAISE EXCEPTION 'La période des inscriptions est déjà passée';
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

-- Création du Trigger associé à la procédure stockée tgSoumissionVoeu
CREATE OR REPLACE TRIGGER tgSoumissionVoeu
    BEFORE INSERT OR UPDATE ON Voeu
    FOR EACH ROW EXECUTE PROCEDURE tgSoumissionVoeu();
```

- L'état d'un voeu (Voeu.état) ne peut pas passer de "refusé" à "validé", ni passer de "refusé" ou "validé" à "déposé" : Dynamique Forte

```
-- L'état d'un voeu (Voeu.état) ne peut pas passer de "refusé" à "validé" ; ni passer de "refusé" ou "validé" à "déposé" : Dynamique Forte

CREATE OR REPLACE FUNCTION tgMajEtatVoeu() RETURNS TRIGGER AS $$

BEGIN
    -- On lève des exceptions si la contrainte n'est pas respectée
    IF OLD.etat = 'refusé' AND NEW.etat = 'validé' THEN
        RAISE EXCEPTION 'L''état d''un voeu ne peut pas passer de refusé à validé';
    ELSEIF OLD.etat = 'refusé' AND NEW.etat = 'déposé' THEN
        RAISE EXCEPTION 'L''état d''un voeu ne peut pas passer de refusé à déposé';
    ELSEIF OLD.etat = 'validé' AND NEW.etat = 'déposé' THEN
        RAISE EXCEPTION 'L''état d''un voeu ne peut pas passer de déposé à déposé';
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

-- Création du Trigger associé à la procédure stockée tgMajEtatVoeu
CREATE OR REPLACE TRIGGER tgMajEtatVoeu
    BEFORE INSERT OR UPDATE ON Voeu
    FOR EACH ROW EXECUTE PROCEDURE tgMajEtatVoeu();
```

- La priorité d'un vœu appartient nécessairement à l'ensemble {1, 2, 3} : Statique Forte.

```sql
CREATE TABLE Voeu(
    idVoeu SERIAL,
    prio INTEGER NOT NULL CHECK (prio >= 1 AND prio <= 3), --contrainte prio entre 1 et 3
    etat EEtatVoeu NOT NULL DEFAULT 'déposé',
    idEtablissement INTEGER NOT NULL REFERENCES Etablissement,
    idOuvrage INTEGER NOT NULL REFERENCES Ouvrage,
    idRef INTEGER NOT NULL REFERENCES Referent,
    idEdition INTEGER NOT NULL REFERENCES Edition,
    idStatsCV INTEGER NULL REFERENCES StatsCampagneVoeux,
    PRIMARY KEY (idVoeu)
);
```

Pour la classe **Intervention** :

- Intervention.date est compris entre les dates Edition.débutFestival et Edition.finFestival : Statique Forte
- L'heure de début d'une intervention (Intervention.hDébut) est nécessairement antérieure à son heure de fin (Intervention.hFin) : Statique Forte
- nbEleves est positif ou nul : Statique Forte
- une intervention dure entre 2 et 4h : Statique Forte

Ces contraintes sont implantées dans la table directement :

```sql
CREATE TABLE Intervention(
    idInterv SERIAL,
    etatInterv EEtatInterv NOT NULL DEFAULT 'planifiée',
    dateInterv DATE NOT NULL,
    hDebut TIME(0) NOT NULL, -- (0) = pas de seconde donc hh:mm
    hFin TIME(0) NOT NULL, -- (0) = pas de seconde donc hh:mm
    nbEleves INTEGER CHECK (nbEleves > 0), -- est null tant que l'intervention n'a pas lieu
    idAuteur INTEGER NOT NULL REFERENCES Auteur,
    idInterp INTEGER REFERENCES Interprete, -- peut être null (par défaut en PostgreSQL) si on se fie
    idAcc INTEGER NOT NULL REFERENCES Accompagnateur,
    idEdition INTEGER NOT NULL REFERENCES Edition,
    idEtablissement INTEGER NOT NULL REFERENCES Etablissement,
    idStatsI INTEGER REFERENCES StatsInterventions, -- est null tant que les stats n'ont pas été géné
    PRIMARY KEY (idInterv),
    CHECK (hFin -  hDebut > INTERVAL '2 hours' AND hFin -  hDebut < INTERVAL '4 hours')
);
```

- L’état d’une intervention (Intervention.étatInterv) ne peut pas passer de “annulée” ou “remplacé” à “planifiée”, ni de “planifiée” à “remplacée” : Dynamique Forte

```sql
-- L’état d’une intervention (Intervention.étatInterv) ne peut pas passer de “annulée” ou “remplacé” à “planifiée”, ni de “planifiée” à “remplacée”

CREATE OR REPLACE FUNCTION tgMajEtatInterv() RETURNS TRIGGER AS $$

BEGIN
    -- On lève des exceptions si la contrainte n'est pas respectée
    IF OLD.etat = 'annulée' AND NEW.etat = 'planifiée' THEN
        RAISE EXCEPTION 'L''état d''une intervention ne peut pas passer de annulée à planifiée';
    ELSEIF OLD.etat = 'remplacée' AND NEW.etat = 'planifiée' THEN
        RAISE EXCEPTION 'L''état d''une intervention ne peut pas passer de remplacée à planifiée';
    ELSEIF OLD.etat = 'planifiée' AND NEW.etat = 'remplacée' THEN
        RAISE EXCEPTION 'L''état d''une intervention ne peut pas passer de planifiée à remplacée';
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

-- Création du Trigger associé à la procédure stockée tgMajEtatInterv
CREATE OR REPLACE TRIGGER tgMajEtatInterv
    BEFORE INSERT OR UPDATE ON Intervention
    FOR EACH ROW EXECUTE PROCEDURE tgMajEtatInterv();
```

- un auteur est limité à 3 interventions par jour : Statique Forte

```sql
-- un auteur est limité à 3 interventions par jour : Statique Forte (/!\ remplacements)

CREATE OR REPLACE FUNCTION tg3IntervJour() RETURNS TRIGGER AS $$
DECLARE
    v_nbInterv INTEGER; -- variable qui stock le nombre d'interventions fait par l'auteur de l'intervention le jour de l'intervention
BEGIN
    -- Récupération du nombre d'interventions fait par l'auteur de l'intervention le jour de l'intervention
    SELECT COUNT(*) INTO v_nbInterv FROM Intervention i
        WHERE NEW.idAuteur = i.idAuteur
        AND NEW.idDateInterv = i.idDateInterv;
    -- On lève des exceptions si la contrainte n'est pas respectée
    IF v_nbInterv >= 3 THEN
        RAISE EXCEPTION 'L''auteur ne peut pas faire plus d''interventions (>3) ce jour';
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

-- Création du Trigger associé à la procédure stockée tg3IntervJour
CREATE OR REPLACE TRIGGER tg3IntervJour
    BEFORE INSERT OR UPDATE ON Intervention
    FOR EACH ROW EXECUTE PROCEDURE tg3IntervJour();
```

Pour la classe **StatsCampagneVoeux** :

- annee doit être antérieure ou égale à l’année en cours : Statique Forte
- tauxParticipation est compris entre 0 et 1 : Statique Forte
- nbVoeuSoumis est supérieur ou égal à nbEtaParticipants qui est lui même supérieur ou egal à nbEtaVoeuSoumis : Statique Forte

Ces contraintes sont implantées dans la table directement :

```sql
CREATE TABLE StatsCampagneVoeux( -- générées à la fin de la campagne de voeux
    idStatsCV SERIAL,
    annee INTEGER UNIQUE NOT NULL CHECK (annee > 2000 AND annee < 2100),
    nbEtaParticipants INTEGER NOT NULL CHECK (nbEtaParticipants > 0),
    nbEtaVoeuSoumis INTEGER NOT NULL CHECK (nbEtaVoeuSoumis > 0),
    nbVoeux INTEGER NOT NULL CHECK (nbVoeux > 0),
    nbVoeuxMoyen NUMERIC(2,2) GENERATED ALWAYS AS (nbVoeux / nbEtaParticipants) STORED,
    tauxParticipation NUMERIC(2,2) GENERATED ALWAYS AS (nbEtaVoeuSoumis / nbEtaParticipants) STORED, --marche comme DEFAULT mais maj à chaque modif
    PRIMARY KEY (idStatsCV),
    CHECK (nbEtaParticipants >= nbEtaVoeuSoumis),
    CHECK (nbEtaVoeuSoumis >= nbVoeux)
);
```

Pour la classe **StatsInterventions** :

- annee doit être antérieure ou égale à l'année en cours : Statique Forte
- nbElevesMoyen est supérieur ou égal à 0 : Statique Forte
- nbInterventions est inférieur ou égale à nbElevesCumulés : Statique Forte

Ces contraintes sont implantées dans la table directement :

```
CREATE TABLE StatsInterventions(
    idStatsI SERIAL,
    annee INTEGER UNIQUE NOT NULL CHECK (annee > 2000 AND annee < 2100),
    nbElevesCumule INTEGER NOT NULL CHECK (nbElevesCumule > 0),
    nbInterventions INTEGER NOT NULL CHECK (nbInterventions > 0),
    nbElevesMoyen NUMERIC(5,2) GENERATED ALWAYS AS (nbElevesCumule / nbInterventions) STORED, --marche comme DEFAULT mais maj à chaque modif
    PRIMARY KEY (idStatsI),
    CHECK (nbInterventions <= nbElevesCumule)
);
```

Pour la classe **StatsOuvrages** :

- nbOuvrages, nbAuteur, nbOuvragesEnfant, nbOuvragesAdultes sont tous positifs ou égale à 0 : Statique Forte

Cette contrainte est implantée directement dans la table :

```
CREATE TABLE StatsOuvrages( -- générées à la fin des inscriptions
    idStatsO SERIAL,
    annee INTEGER UNIQUE NOT NULL CHECK (annee > 2000 AND annee < 2100),
    nbOuvrages INTEGER NOT NULL CHECK (nbOuvrages >= 0),    -- tous les "nb" sont positifs
    nbAuteurs INTEGER NOT NULL CHECK (nbAuteurs >= 0),
    nbOuvragesEnfant INTEGER NOT NULL CHECK (nbOuvragesEnfant >= 0),
    nbOuvragesAdulte INTEGER NOT NULL CHECK (nbOuvragesAdulte >= 0),
    PRIMARY KEY (idStatsO)
);
```

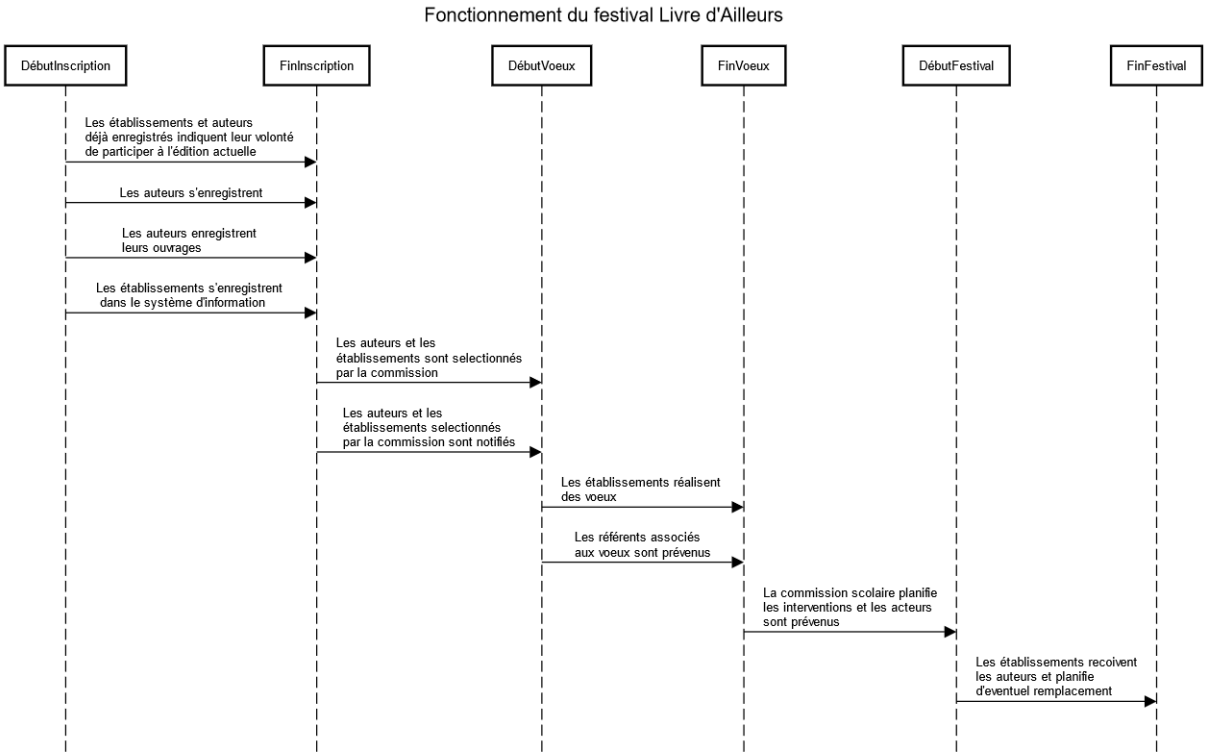
Pour la classe **Interprète** :

- langueSource est différente de langueCible : Statique Forte

Cette contrainte est implantée dans la table directement :

```
CREATE TABLE Interprete(
    idInterp SERIAL,
    nom VARCHAR(30) NOT NULL,
    prenom VARCHAR(30) NOT NULL,
    numTel VARCHAR(12) UNIQUE CHECK (LENGTH(numTel) >= 10), -- donc taille entre 10 et 12 (on peut ajouter +..
    mail EmailAddress UNIQUE,
    mdp MotDePasse,
    langueSource VARCHAR(30) NOT NULL,
    langueCible VARCHAR(30) NOT NULL,
    PRIMARY KEY (idInterp),
    UNIQUE (nom, prenom), -- couple nom, prénom unique
    CHECK (langueSource != langueCible) -- la langue source et la langue cible doivent être différents
);
```

Fonctionnement d’une édition d’un festival (intéractions & déroulement)



Description de l'implantation (application web)

Structure du projet

Le projet est structuré dans deux dossiers : backend et frontend. Les scripts de création de la base de données sont dans le dossier :

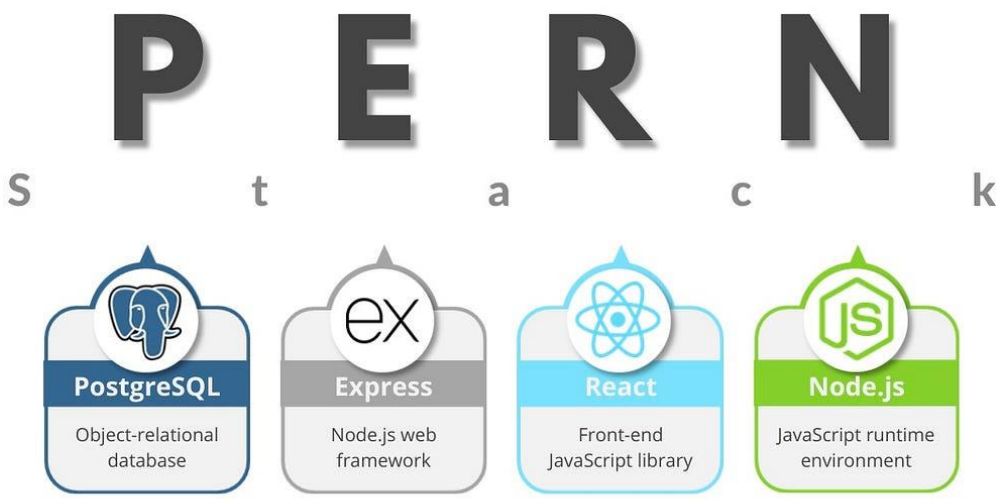
LIVRE_AILLEURS\backend\db\Scripts_crea_db

Pour créer la db, veillez à executer les scripts dans le bon ordre :

Domain → Type → Table → Procédures → Fonctions

Stack : description des frameworks

Pour réaliser cette application web, nous avons choisi le stack PERN :



Source de l'image : source

Une application basée sur le stack PERN (PostgreSQL, Express, React, Node.js) est une architecture full-stack moderne conçue pour développer des applications web robustes et évolutives. Chaque composant du stack joue un rôle spécifique dans le développement et le fonctionnement de l'application. Voici comment ces composants interagissent typiquement :

1. **PostgreSQL (Base de données relationnelle) :**
    - PostgreSQL est un système de gestion de base de données relationnelle open source, réputé pour sa fiabilité et ses fonctionnalités avancées.
    - Dans une application PERN, PostgreSQL est utilisé pour stocker et organiser les données de manière structurée.

- Les schémas, tables et relations sont définis dans PostgreSQL pour représenter les différentes entités et leurs interactions dans l'application.

2. **Express (Framework côté serveur) :**
    - Express est un framework web minimaliste et flexible pour Node.js, permettant de construire des applications web et des API robustes.
    - Dans une application PERN, Express est utilisé pour créer et gérer les routes HTTP, les requêtes et les réponses entre le serveur et le client.
    - Il facilite également la gestion des middlewares pour l'authentification, la validation des données, la gestion des erreurs, etc.
3. **React (Bibliothèque côté client) :**
    - React est une bibliothèque JavaScript développée par Facebook, utilisée pour construire des interfaces utilisateur interactives et dynamiques.
    - Dans une application PERN, React est utilisé pour créer l'interface utilisateur côté client.
    - Il permet de diviser l'interface utilisateur en composants réutilisables, ce qui facilite la gestion de l'état de l'application et le rendu des données provenant du serveur.
4. **Node.js (Environnement d'exécution côté serveur) :**
    - Node.js est un environnement d'exécution JavaScript côté serveur, construit sur le moteur V8 de Chrome.
    - Dans une application PERN, Node.js est utilisé pour exécuter le serveur web et gérer les opérations côté serveur, telles que l'accès à la base de données, la manipulation des requêtes HTTP, etc.
    - Il permet également de créer des API RESTful pour communiquer avec le client et de gérer les opérations asynchrones de manière efficace grâce à son modèle basé sur les événements.

Nous utilisons TailwindCSS pour styliser les composants (notamment les formulaires). Tailwind CSS est un framework CSS utilitaire qui permet de styliser rapidement et efficacement les éléments HTML en utilisant des classes prédéfinies directement dans le code HTML ou JSX. Contrairement aux frameworks traditionnels, il favorise une approche basée sur des utilitaires CSS pour personnaliser les styles de manière concise et cohérente.

Fonctionnement typique de l’application :

1. Lorsqu'un utilisateur interagit avec l'application, une requête est envoyée depuis le navigateur vers le serveur Express via une URL spécifique.
2. Express reçoit la requête et la dirige vers la route appropriée, en fonction de la méthode HTTP (GET, POST, PUT, DELETE) et de l'URL demandée.
3. Si la route nécessite des données de la base de données, Express utilise un module PostgreSQL pour exécuter les requêtes SQL correspondantes et récupérer les données demandées.
4. Une fois les données récupérées, Express les formate et les envoie en réponse au client sous forme de JSON.
5. Le client React reçoit les données et les utilise pour mettre à jour l'interface utilisateur, en affichant les informations pertinentes dans les composants appropriés.
6. Si l'utilisateur effectue une action qui nécessite une mise à jour des données (par exemple, la soumission d'un formulaire), React envoie une requête au serveur Express via une requête HTTP.
7. Express traite la requête, effectue les modifications nécessaires dans la base de données PostgreSQL et renvoie une réponse indiquant le succès ou l'échec de l'opération.
8. En fonction de la réponse reçue, React met à jour l'interface utilisateur pour refléter les changements effectués dans la base de données.

En résumé, le stack PERN permet de créer des applications web dynamiques en combinant une base de données relationnelle robuste (PostgreSQL), un serveur web léger et flexible (Express), une interface utilisateur réactive et modulaire (React) et un environnement d'exécution côté serveur performant (Node.js). Cette architecture favorise la modularité, la scalabilité et la maintenabilité des applications tout en offrant une expérience utilisateur fluide et interactive.

Installation de notre application

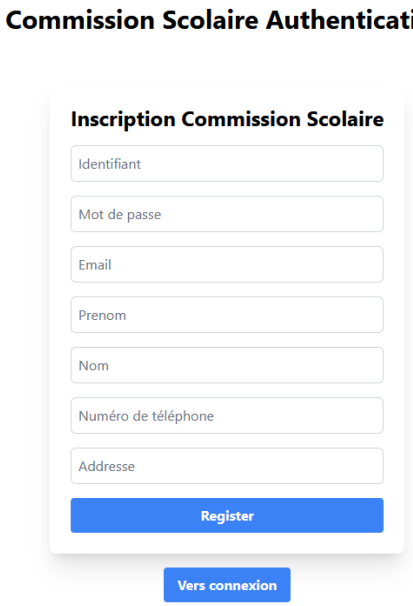
L'entierté du code de l'application ainsi que les fichiers sql pour créer la base de données se trouve sur GitHub à l'adresse : dépôt github LIVREDAILLEURS. Vous trouverez aussi une copie des fichiers dans l'archive zip fournie. Vous trouverez un fichier README.md sur GitHub vous guidant dans l'installation, le processus est simple mais nécessite d'avoir Postgresql et Node.js installé sur votre machine.

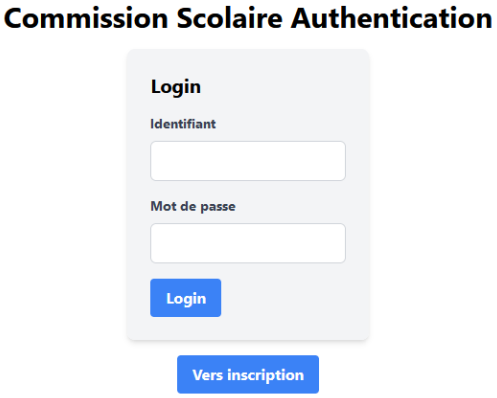
Notre application

Au lancement de l'application, on arrive sur la homepage :

En fonction de l'utilisateur que vous êtes, vous cliquez sur le bouton correspondant.

Dans les trois cas, cela vous redirige vers une page d'authentification (inscription et connexion) :



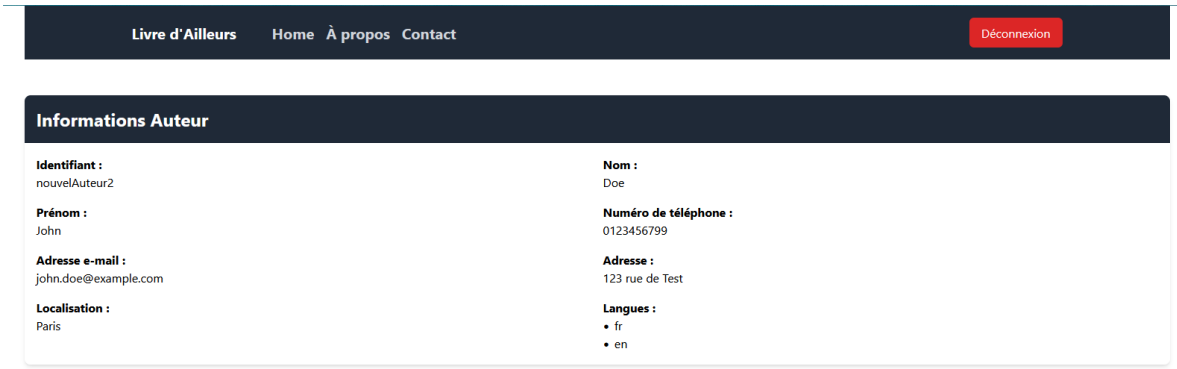


Les pages d'authentification sont semblables pour les auteurs et les établissements.

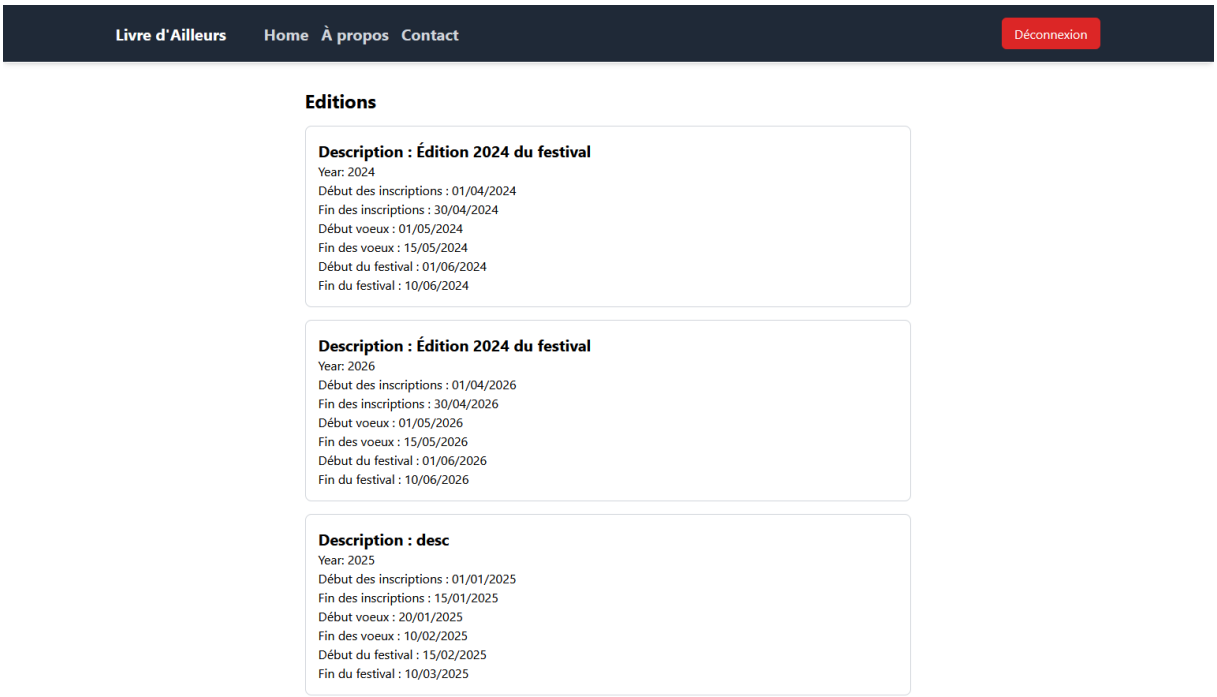
Après une inscription puis une connexion ou simplement une connexion dans le cas ou vous êtes déjà enregistré, vous arrivez sur le dashboard spécifique à votre type d'utilisateur. Ce dashboard centralise toutes les fonctionnalités disponibles pour le type d'acteur en question. Voici comment sont composés les dashboard :

Dashboard auteur

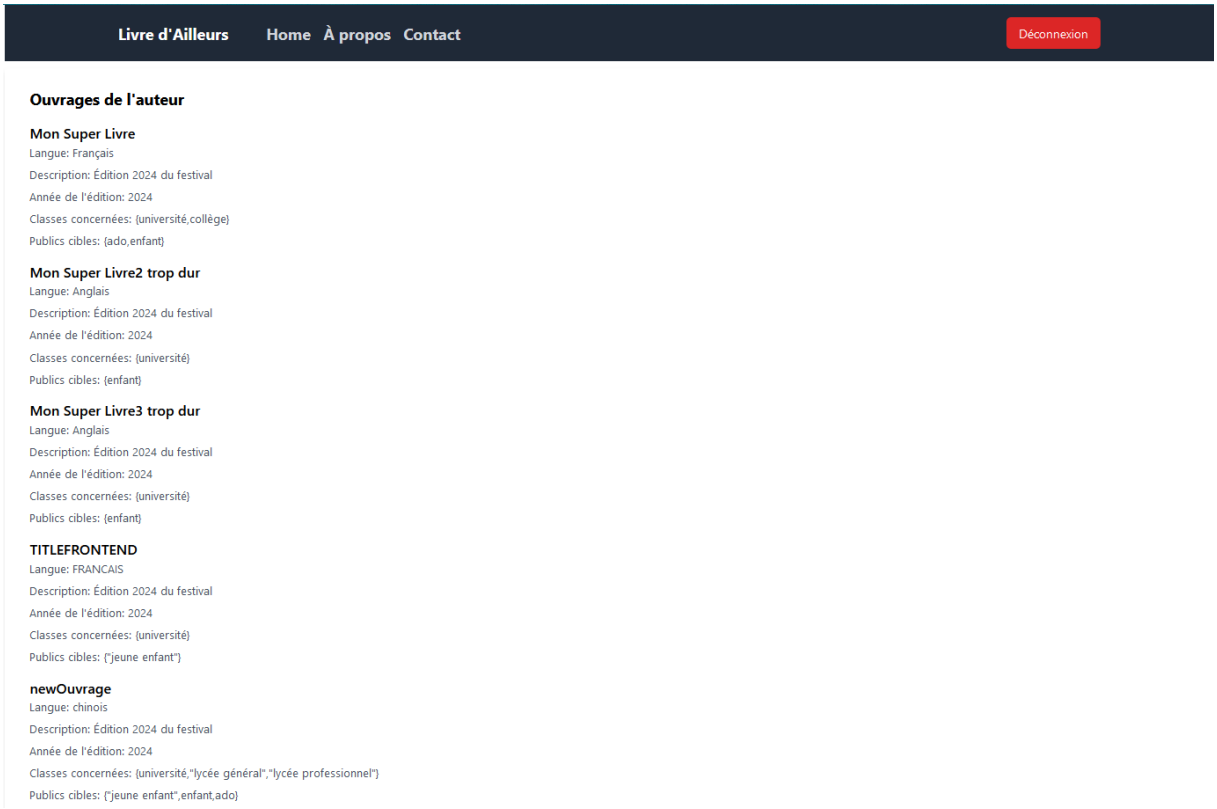
D'abord une partie qui rappelle les informations spécifiques de l'auteur :



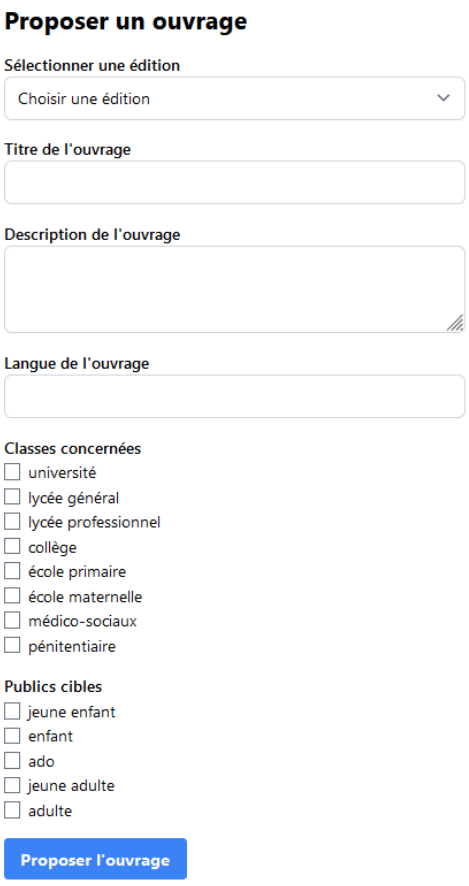
Ensuite, une partie qui affichent toutes les éditions de festival :



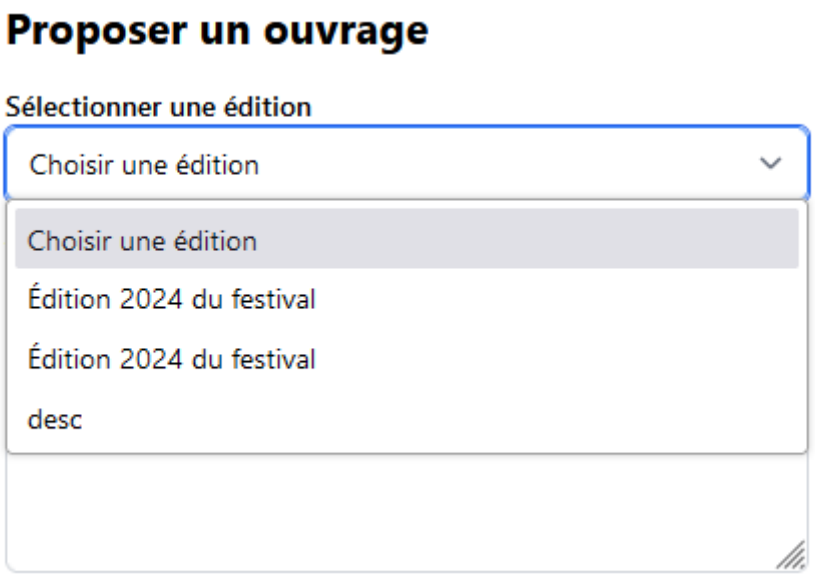
Puis, une partie qui montre tous les ouvrages proposés par l'auteur :



Enfin, un formulaire pour proposer un ouvrage :

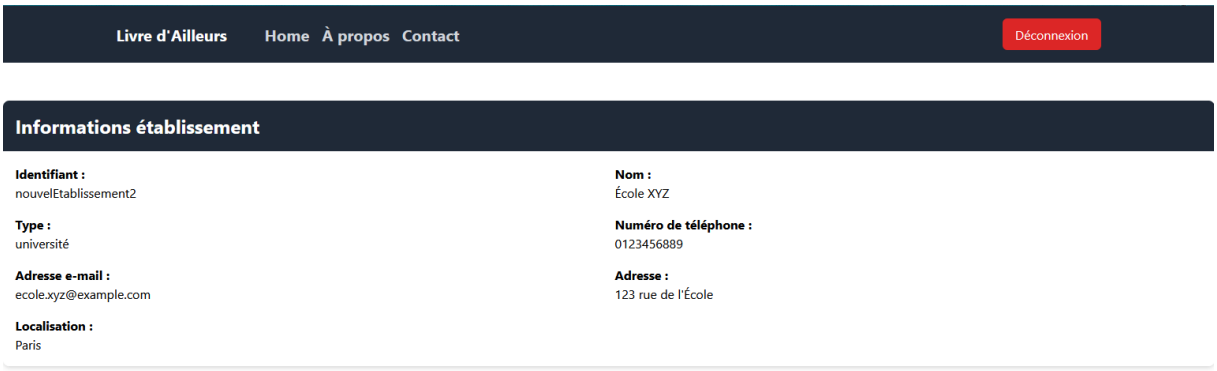


Tous ces affichages sont dynamiques. Par exemple, lors de la proposition d'un ouvrage, l'auteur n'a accès qu'aux editions existantes :

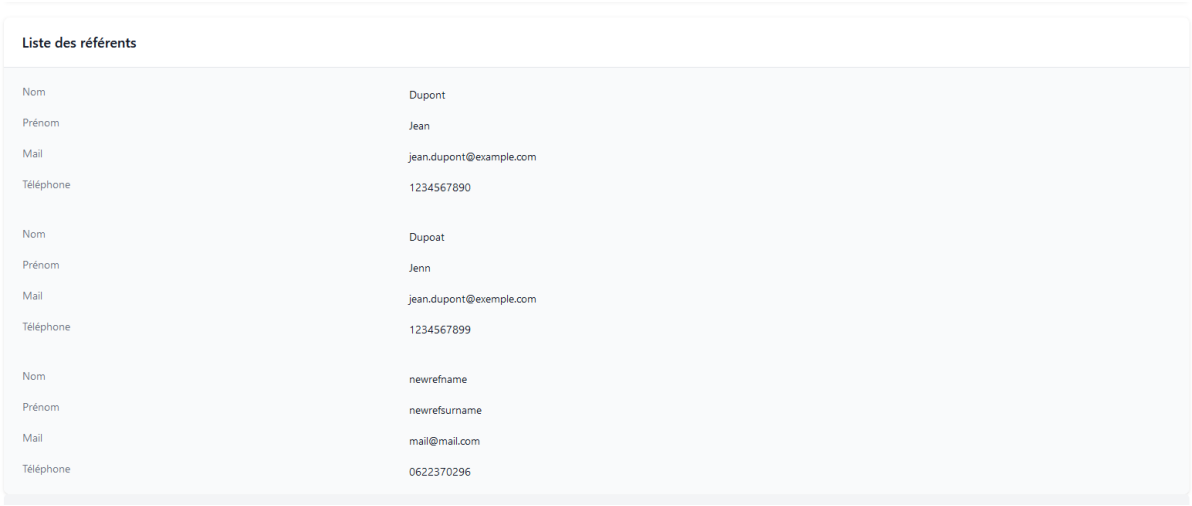


Dashboard établissement

D’abord une partie qui rappelle les informations spécifiques de l’établissement :

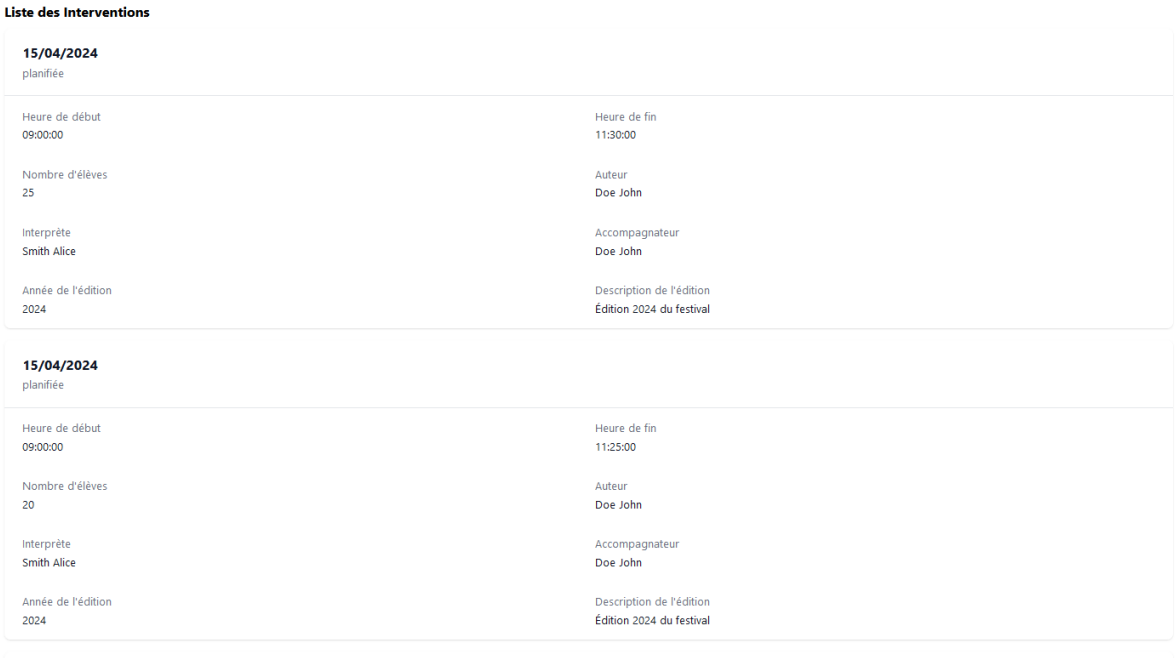


Puis la liste des référents liés à cet établissement :

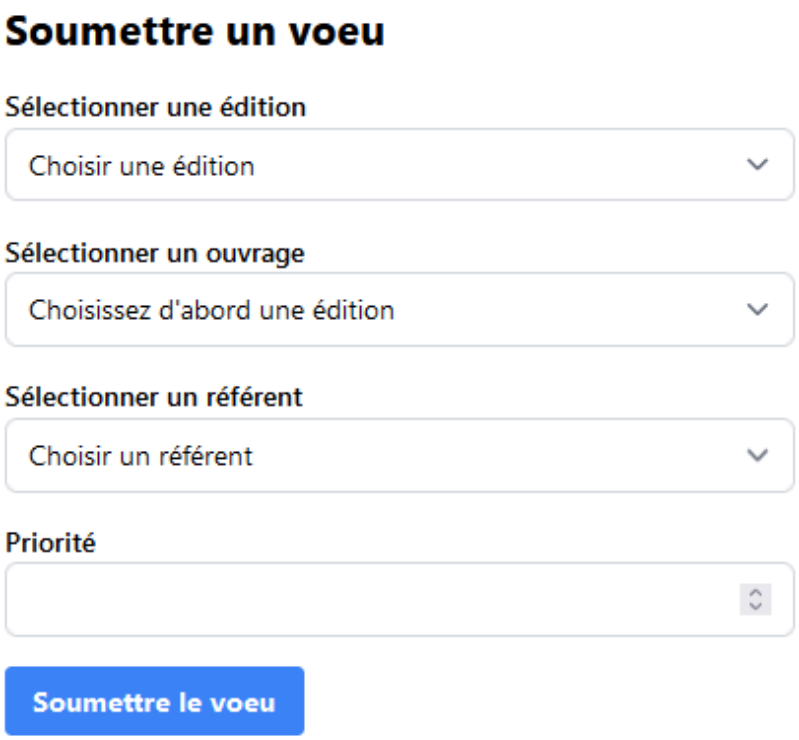


Un formulaire pour rajouter un référent :

La liste des interventions prévues pour cet établissement :



Enfin, le formulaire pour soumettre un vœu :



Ce composant est dynamique, c'est-à-dire qu'on choisit d'abord une édition et ensuite les ouvrages dans la liste déroulante proposés sont propres à cette édition :

Soumettre un voeu

Sélectionner une édition

Édition 2024 du festival

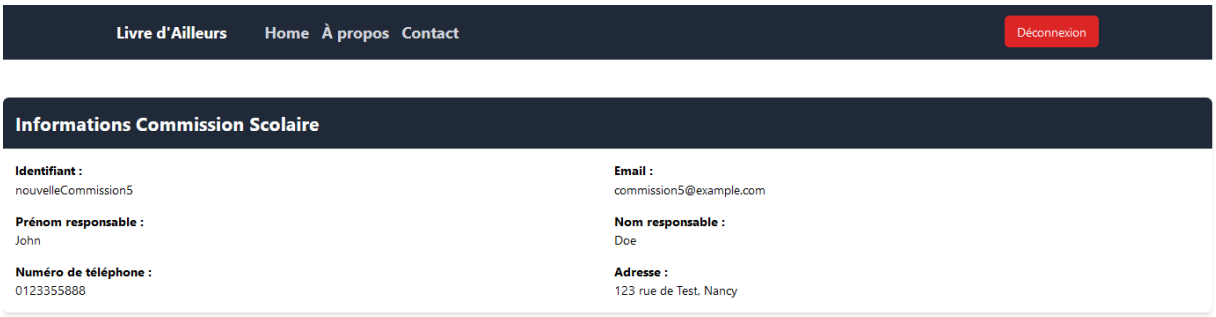
Sélectionner un ouvrage

Choisissez d'abord une édition

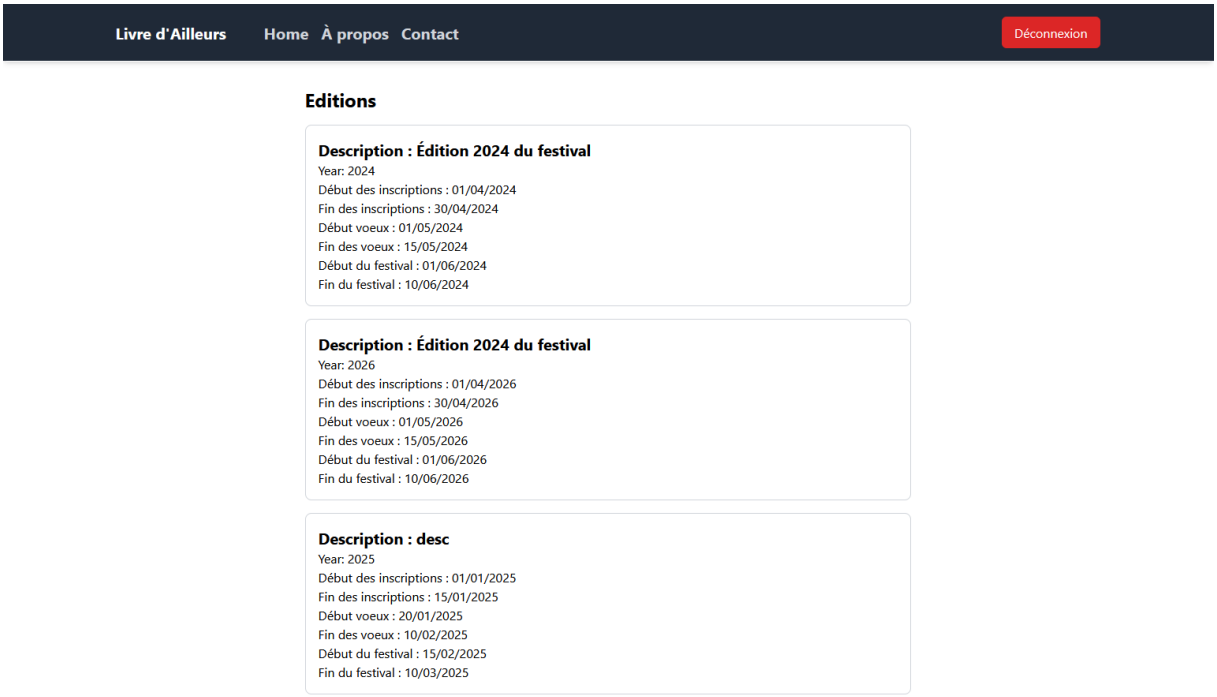
- Choisissez d'abord une édition
- Mon Super Livre
- Mon Super Livre2 trop dur
- Mon Super Livre3 trop dur
- TITLEFRONTEND
- newOuvrage
- titreTEST
- newtest

Dashboard commission scolaire

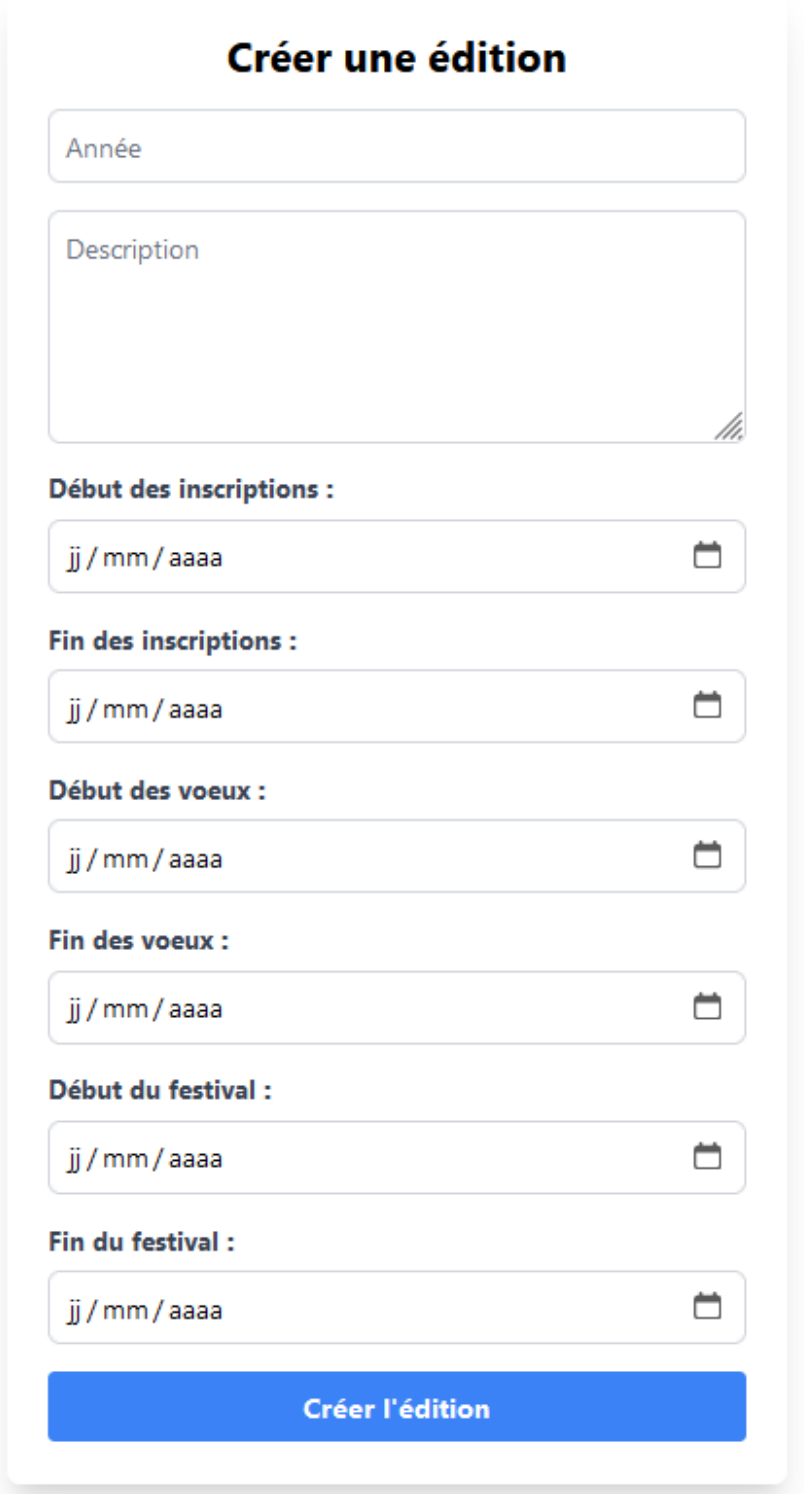
D’abord une section qui rappelle les informations spécifiques de la commission scolaire :



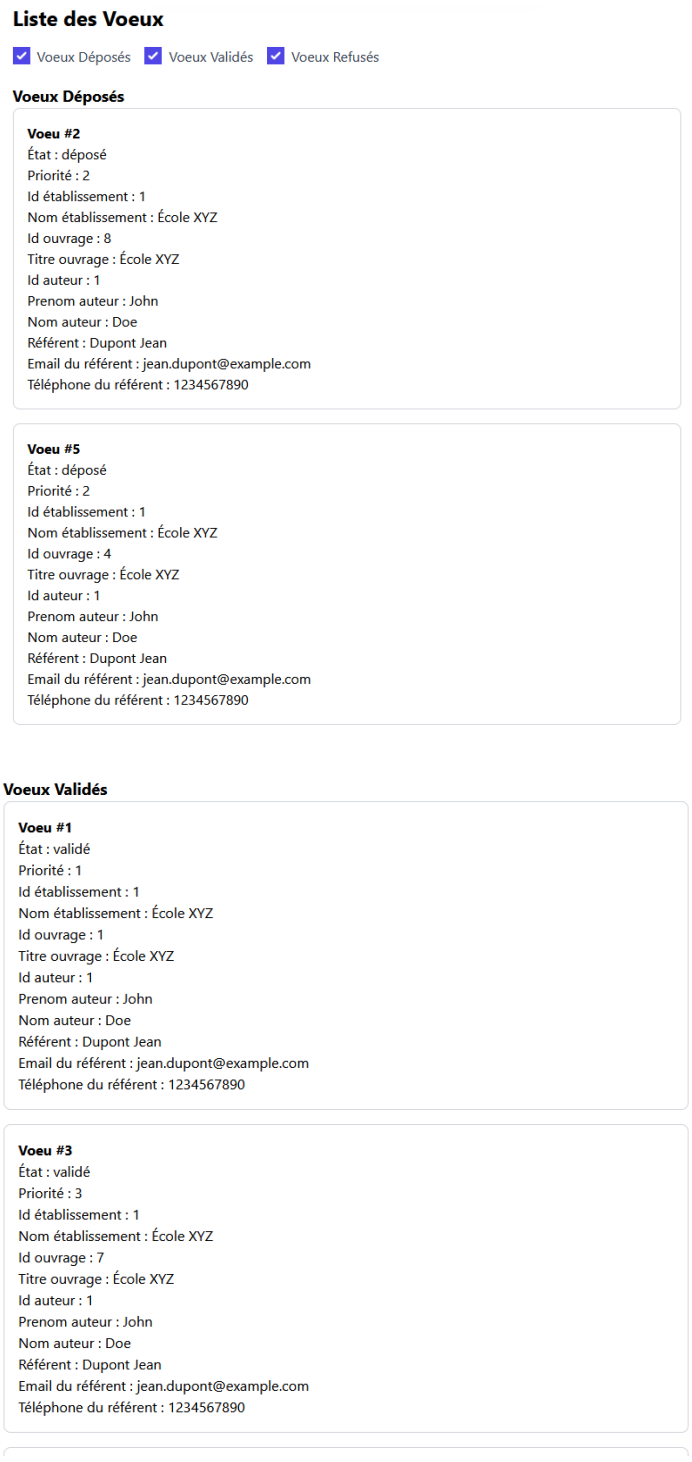
Ensuite, une partie qui affichent toutes les éditions de festival :



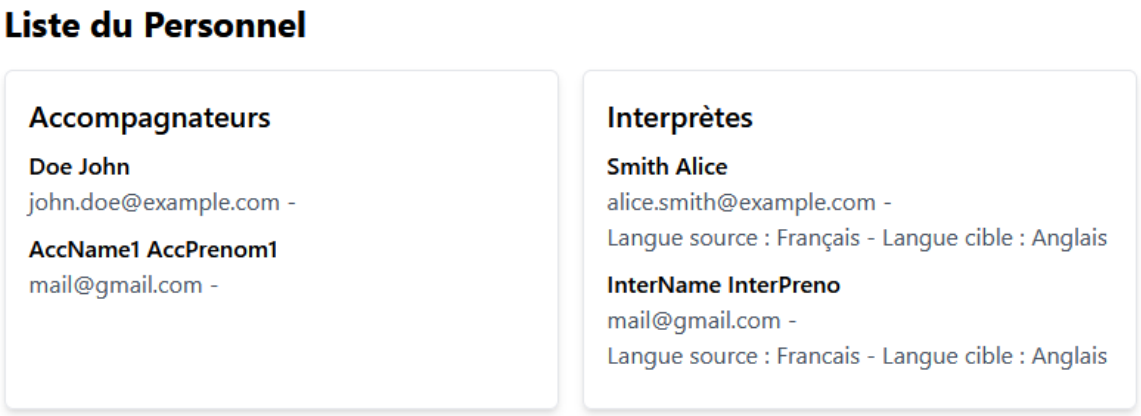
Puis, un formulaire pour créer une nouvelle édition de festival :



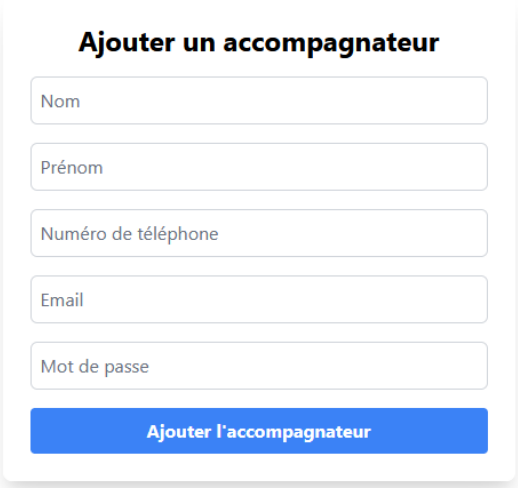
Après, une section qui montre la liste des vœux des établissements :



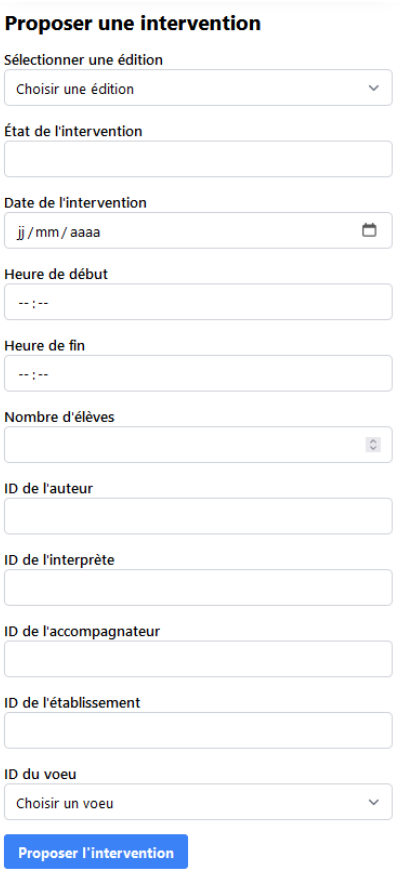
Un composant qui montre la liste du personnel :



Deux formulaires permettant de renseigner des nouveaux accompagnateurs et interprètes :



Enfin, un formulaire permettant de soumettre une intervention :



Tous ces affichages sont bien évidemment dynamiques et les informations sont récupérées en temps réel.

Fonctionnement technique

Il serait trop long (et pas très intéressant) de décrire le fonctionnement de chaque composant et de chaque route. Je vais donc présenter trois routes importantes.

Routes pour l'authentification

Nous utilisons les JSON Web Token pour assurer l'authentification. Un JSON Web Token (JWT) est un standard permettant de sécuriser les échanges d'informations entre parties. Il est composé de trois parties : l'en-tête, le payload et la signature.

Dans le contexte de l'authentification, après qu'un utilisateur se soit connecté avec succès, un JWT est généré côté serveur. Ce jeton est ensuite inclus dans un cookie HTTP qui est stocké côté client. Lorsque l'utilisateur effectue des requêtes ultérieures vers le serveur, le JWT est automatiquement inclus dans l'en-tête HTTP Authorization ou dans le cookie, permettant au serveur de vérifier l'identité de l'utilisateur à chaque requête sans avoir besoin de maintenir un état de session côté serveur. Cette méthode d'authentification sans état (stateless) simplifie la gestion des sessions côté serveur et permet une meilleure évolutivité de l'application.

Route pour l'inscription

Voici la route pour l'inscription des auteurs :

```
// Route pour l'inscription des auteurs
router.post('/register/auteur', async (req, res) => {
    const { identifiant, mdp, nom, prenom, num_tel, mail, adresse, localisation, langues } = req.body;
    try {
        // Hasher le mot de passe avec bcrypt
        const hashedPassword = await bcrypt.hash(mdp, saltRounds);

        // Insérer l'auteur dans la base de données
        await db.query('CALL sp_insert_auteur($1, $2, $3, $4, $5, $6, $7, $8, $9)', [identifiant, hashedPassword, nom, prenom, num_tel, mail, adresse, localisation, langues]);

        // Répondre avec un message de succès
        res.status(201).json({ message: 'Auteur inscrit avec succès' });
    } catch (error) {
        console.error('Error during registration:', error);
        res.status(500).json({ message: 'Erreur serveur' });
    }
});
```

Ce code représente une route d'inscription pour les auteurs dans une application web. Voici ce que chaque partie du code fait :

1. **Définition de la route :**
    - La route est définie pour la méthode HTTP POST à l'URI '/register/auteur'.
2. **Fonction de gestion de la requête :**
    - Lorsqu'une requête POST est effectuée sur '/register/auteur', la fonction de gestion de la requête est déclenchée.
    - Cette fonction prend deux paramètres, req (requête) et res (réponse), qui représentent respectivement l'objet de requête et l'objet de réponse HTTP.

3. **Extraction des données de la requête :**
    - Les données envoyées avec la requête sont extraites du corps de la requête (req.body). Ces données comprennent identifiant, mdp (mot de passe), nom, prenom, num_tel (numéro de téléphone), mail (adresse e-mail), adresse, localisation et langues.
4. **Hachage du mot de passe :**
    - Le mot de passe fourni est haché à l'aide de l'algorithme de hachage bcrypt, afin de sécuriser le stockage du mot de passe dans la base de données. Le résultat haché est stocké dans la variable hashedPassword.
5. **Insertion dans la base de données :**
    - Les données de l'auteur, y compris l'identifiant, le mot de passe haché et d'autres informations, sont insérées dans la base de données à l'aide d'une requête SQL. La méthode db.query est utilisée pour appeler une procédure stockée (sp_insert_auteur) avec les paramètres nécessaires.
6. **Réponse à la requête :**
    - Si l'inscription est effectuée avec succès, une réponse HTTP avec le code de statut 201 (Created) est renvoyée, accompagnée d'un objet JSON contenant un message de succès.
    - En cas d'erreur lors de l'inscription (par exemple, si une erreur survient lors du hachage du mot de passe ou lors de l'insertion dans la base de données), une réponse avec le code de statut 500 (Internal Server Error) est renvoyée, accompagnée d'un message d'erreur générique.

En résumé, cette route permet à un utilisateur de s'inscrire en tant qu'auteur dans l'application. Les données fournies sont stockées de manière sécurisée dans la base de données après avoir été hachées, et une réponse appropriée est renvoyée au client en fonction du résultat de l'opération d'inscription.

Route pour la connexion

Similairement, voici la route pour la connexion des auteurs :

```
// Route pour le login des auteurs
router.post('/login/auteur', async (req, res) => {
    const { identifiant, mdp } = req.body;
    try {
        // Récupérer l'auteur correspondant à l'identifiant
        const result = await db.query('SELECT * FROM Auteur WHERE identifiant = $1', [identifiant]);

        // Vérifier si l'auteur existe
        if (result.rows.length === 0) {
            return res.status(404).json({ message: 'Auteur non trouvé' });
        }

        // Récupérer les données de l'auteur
        const auteur = result.rows[0];

        // Vérifier si le mot de passe est correct en comparant avec le hash stocké
        const passwordMatch = await bcrypt.compare(mdp, auteur.mdp);
        if (!passwordMatch) {
            return res.status(401).json({ message: 'Mot de passe incorrect' });
        }

        // Créer un token JWT pour l'auteur
        const token = jwt.sign({ id: auteur.idauteur, role: 'auteur' }, jwtSecret);

        // Répondre avec le token JWT
        res.cookie('token', token, { httpOnly: false, sameSite: 'Lax' }).json({ idauteur : auteur.idauteur });
    } catch (error) {
        console.error('Error during login:', error);
        res.status(500).json({ message: 'Erreur serveur' });
    }
});
```

Cette route représente le processus de connexion d'un auteur dans l'application. Voici une explication détaillée du fonctionnement de chaque partie du code :

1. **Définition de la route :**
    - Cette route est définie pour la méthode HTTP POST à l'URI '/login/auteur'.
2. **Fonction de gestion de la requête :**
    - Lorsqu'une requête POST est effectuée sur '/login/auteur', la fonction de gestion de la requête est déclenchée.
    - Cette fonction prend deux paramètres, req (requête) et res (réponse), qui représentent respectivement l'objet de requête et l'objet de réponse HTTP.
3. **Extraction des données de la requête :**
    - Les données envoyées avec la requête sont extraites du corps de la requête (req.body). Ces données comprennent identifiant et mdp (mot de passe).
4. **Récupération de l'auteur correspondant à l'identifiant :**
    - Une requête SQL est envoyée à la base de données pour récupérer les données de l'auteur correspondant à l'identifiant fourni. La méthode db.query est utilisée pour exécuter la requête SQL.
5. **Vérification de l'existence de l'auteur :**

- Si aucun auteur correspondant à l'identifiant n'est trouvé dans la base de données, une réponse avec le code de statut 404 (Not Found) est renvoyée, indiquant que l'auteur n'a pas été trouvé.

6. **Vérification du mot de passe :**
   - Si un auteur correspondant à l'identifiant est trouvé, le mot de passe fourni est comparé avec le hash stocké dans la base de données à l'aide de la fonction bcrypt.compare.
   - Si le mot de passe ne correspond pas au hash stocké, une réponse avec le code de statut 401 (Unauthorized) est renvoyée, indiquant que le mot de passe est incorrect.
7. **Création d'un token JWT :**
   - Si le mot de passe correspond au hash stocké, un token JWT (JSON Web Token) est créé pour l'auteur à l'aide de la fonction jwt.sign.
   - Le token JWT contient l'identifiant de l'auteur et son rôle (dans ce cas, 'auteur'), ainsi qu'une signature cryptographique pour garantir son authenticité.
8. **Réponse à la requête avec le token JWT :**
   - Le token JWT est inclus dans un cookie HTTP, puis une réponse JSON est renvoyée au client avec l'identifiant de l'auteur.
   - Le cookie est configuré avec les options httpOnly: false et sameSite: 'Lax', ce qui signifie qu'il peut être accédé par JavaScript du côté client et qu'il sera envoyé avec les requêtes de navigation croisée, respectivement.
9. **Gestion des erreurs :**
   - Si une erreur survient pendant le processus de connexion (par exemple, une erreur de requête SQL ou une erreur de hachage), une réponse avec le code de statut 500 (Internal Server Error) est renvoyée, accompagnée d'un message d'erreur générique.

En résumé, cette route permet à un auteur de se connecter à l'application en vérifiant son identifiant et son mot de passe. Si les informations d'identification sont correctes, un token JWT est créé et renvoyé au client pour authentifier l'auteur lors des requêtes ultérieures.

Les routes pour l’authentificaiton des autres acteurs sont similaires.

Route pour proposer un ouvrage

Voici la route pour proposer un ouvrage :

```
// Route pour proposer un ouvrage
router.post('/edition/:idEdition/ouvrage', authenticateUser, authorizeAuteur, async (req, res) => {
    const { titre, classesConcernees, publicsCibles, description, langue } = req.body;
    const idAuteur = req.user.id; // Récupérer l'ID de l'auteur à partir du token JWT
    const idEdition = req.params.idEdition;

    try {
        // Appel de la procédure stockée pour proposer l'ouvrage
        const result = await db.query('CALL sp_proposer_ouvrage($1, $2, $3, $4, $5, $6, $7, $8, $9)',
            [titre, classesConcernees, publicsCibles, description, langue, idEdition, idAuteur, false, '']);

        // Vérification du résultat de la procédure stockée
        if (result.rows[0].p_success) {
            res.status(201).json({ message: result.rows[0].p_message });
        } else {
            res.status(500).json({ message: result.rows[0].p_message });
        }
    } catch (error) {
        console.error('Error proposing ouvrage:', error);
        res.status(500).json({ message: 'Erreur serveur' });
    }
});
```

Voici une explication détaillée de son fonctionnement :

1. **Définition de la route :**
    - Cette route est définie pour la méthode HTTP POST à l'URI '/edition/:idEdition/ouvrage'. Le paramètre :idEdition dans l'URI indique l'identifiant de l'édition pour laquelle l'ouvrage est proposé.
2. **Middleware d'authentification et d'autorisation :**
    - Avant d'accéder à la fonction de gestion de la requête, deux middlewares (authenticateUser et authorizeAuteur) sont exécutés. Ces middlewares sont responsables de l'authentification de l'utilisateur et de l'autorisation pour proposer un ouvrage.
3. **Extraction des données de la requête :**
    - Les données envoyées avec la requête sont extraites du corps de la requête (req.body). Ces données comprennent le titre, les classesConcernees, les publicsCibles, la description, et la langue de l'ouvrage.
4. **Récupération de l'identifiant de l'auteur et de l'édition :**
    - L'identifiant de l'auteur est extrait du JWT stocké dans le cookie de la requête (req.user.id). Cela permet de connaître l'auteur qui propose l'ouvrage.

- L'identifiant de l'édition est extrait des paramètres de l'URI (req.params.idEdition), ce qui spécifie l'édition à laquelle l'ouvrage est proposé.

5. **Appel de la procédure stockée pour proposer l'ouvrage :**
    - Une procédure stockée (sp_proposer_ouvrage) est appelée dans la base de données avec les données de l'ouvrage, l'identifiant de l'édition, l'identifiant de l'auteur, ainsi que d'autres paramètres nécessaires.
    - La procédure stockée est responsable de traiter la proposition de l'ouvrage et de retourner un résultat indiquant le succès ou l'échec de l'opération.
6. **Traitement du résultat de la procédure stockée :**
    - Le résultat de la procédure stockée est vérifié pour déterminer si la proposition de l'ouvrage a réussi ou non.
    - Si la proposition est réussie (déterminé par la valeur de la colonne p_success dans le premier enregistrement du résultat), une réponse avec le code de statut 201 (Created) est renvoyée avec un message indiquant le succès de l'opération.
    - En cas d'échec, une réponse avec le code de statut 500 (Internal Server Error) est renvoyée avec un message indiquant l'échec de l'opération.
7. **Gestion des erreurs :**
    - Si une erreur survient pendant le processus de proposition de l'ouvrage (par exemple, une erreur de requête SQL), une réponse avec le code de statut 500 (Internal Server Error) est renvoyée, accompagnée d'un message d'erreur générique.

En résumé, cette route permet à un auteur authentifié de proposer un ouvrage pour une édition spécifique en fournissant les informations nécessaires. Les données sont ensuite traitées dans la base de données via une procédure stockée, et le résultat de l'opération est renvoyé en réponse à la requête.

Route pour récuperer les interventions d'un établissement dans une édition donnée

Voici la route :

```
// Route pour recuperer les interventions d'un établissement dans une édition donnée
router.get('/edition/:idEdition/etablissements/:idEtablissement/interventions', authenticateUser, authorizeEtablissement, async (req, res) => {
    const idEtablissement = req.params.idEtablissement;
    const idEdition = req.params.idEdition;
    try {
        const interventions = await db.query(`
            SELECT
                Intervention.*,
                Auteur.nom AS nomAuteur, Auteur.prenom AS prenomAuteur,
                Interprete.nom AS nomInterprete, Interprete.prenom AS prenomInterprete,
                Accompagnateur.nom AS nomAccompagnateur, Accompagnateur.prenom AS prenomAccompagnateur,
                Edition.annee AS anneeEdition, Edition.description AS descEdition
            FROM Intervention
            LEFT JOIN Auteur ON Intervention.idAuteur = Auteur.idAuteur
            LEFT JOIN Interprete ON Intervention.idInterp = Interprete.idInterp
            LEFT JOIN Accompagnateur ON Intervention.idAcc = Accompagnateur.idAcc
            LEFT JOIN Edition ON Intervention.idEdition = Edition.idEdition
            WHERE Intervention.idEtablissement = $1 AND Intervention.idEdition = $2
        `, [idEtablissement, idEdition]);
        res.status(200).json({ interventions: interventions.rows });
    } catch (error) {
        console.error('Error fetching interventions for establishment:', error);
        res.status(500).json({ message: 'Erreur serveur' });
    }
});
      You, last week • Implemented : ajout d'un referent, recuperer to…
```

Voici une explication détaillée du fonctionnement de cette route :

1. **Définition de la route :**
    - Cette route est définie pour la méthode HTTP GET à l'URI '/edition/:idEdition/etablissements/:idEtablissement/interventions'. Les paramètres de l'URI sont :idEdition et :idEtablissement, qui représentent respectivement l'identifiant de l'édition et de l'établissement.
2. **Middleware d'authentification et d'autorisation :**
    - Avant d'accéder à la fonction de gestion de la requête, deux middlewares (authenticateUser et authorizeEtablissement) sont exécutés. Ces middlewares sont responsables de l'authentification de l'utilisateur et de l'autorisation pour accéder aux interventions de l'établissement.
3. **Extraction des paramètres de la requête :**
    - Les identifiants de l'établissement (idEtablissement) et de l'édition (idEdition) sont extraits des paramètres de l'URI (req.params.idEtablissement et req.params.idEdition).
4. **Requête SQL pour récupérer les interventions :**
    - Une requête SQL est exécutée pour récupérer les interventions de l'établissement dans l'édition spécifiée.

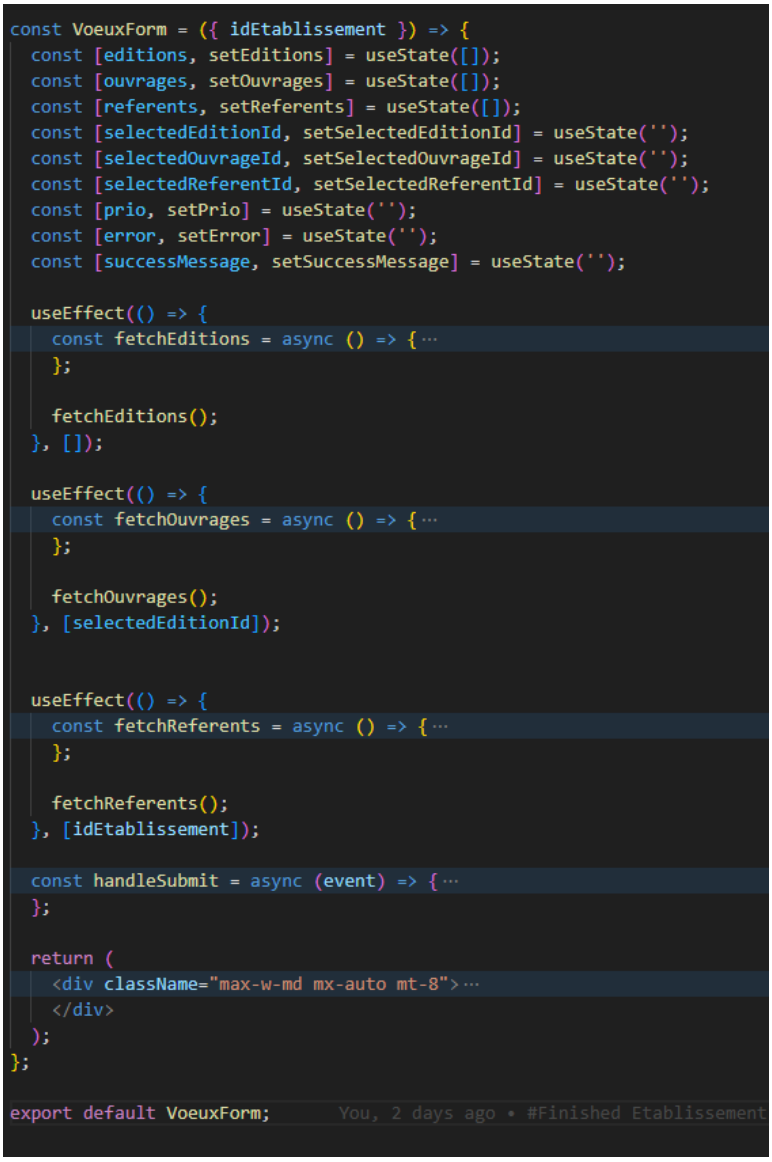
- La requête SQL utilise plusieurs jointures (LEFT JOIN) avec les tables Auteur, Interprete, Accompagnateur et Edition pour récupérer des informations supplémentaires sur les intervenants et l'édition.
- Les informations récupérées comprennent toutes les colonnes de la table Intervention, ainsi que des informations supplémentaires telles que le nom et le prénom des auteurs, interprètes et accompagnateurs, ainsi que l'année et la description de l'édition.

5. **Réponse à la requête :**
   - Les résultats de la requête SQL sont renvoyés en tant que tableau d'objets JSON dans la réponse HTTP, sous la forme d'un objet contenant un tableau nommé interventions.
   - Le code de statut de la réponse est défini à 200 (OK), indiquant que la requête a été traitée avec succès.
6. **Gestion des erreurs :**
   - En cas d'erreur lors de l'exécution de la requête SQL (par exemple, une erreur de syntaxe SQL ou une erreur de base de données), une réponse avec le code de statut 500 (Internal Server Error) est renvoyée, accompagnée d'un message d'erreur générique.

Un composant React en détail

Je vais présenter le composant React qui permet de proposer un vœu :

Voici la structure du composant :



```
const VoeuxForm = ({ idEtablissement }) => {
  const [editions, setEditions] = useState([]);
  const [ouvrages, setOuvrages] = useState([]);
  const [referents, setReferents] = useState([]);
  const [selectedEditionId, setSelectedEditionId] = useState('');
  const [selectedOuvrageId, setSelectedOuvrageId] = useState('');
  const [selectedReferentId, setSelectedReferentId] = useState('');
  const [prio, setPrio] = useState('');
  const [error, setError] = useState('');
  const [successMessage, setSuccessMessage] = useState('');

  useEffect(() => {
    const fetchEditions = async () => {···
    };

    fetchEditions();
  }, []);

  useEffect(() => {
    const fetchOuvrages = async () => {···
    };

    fetchOuvrages();
  }, [selectedEditionId]);


  useEffect(() => {
    const fetchReferents = async () => {···
    };

    fetchReferents();
  }, [idEtablissement]);

  const handleSubmit = async (event) => {···
  };

  return (
    <div className="max-w-md mx-auto mt-8">···
    </div>
  );
};

export default VoeuxForm;      You, 2 days ago • #Finished Etablissement
```

D'abord on récupère les éditions :

```
useEffect(() => {
  const fetchEditions = async () => {
    try {
      const response = await axios.get('http://localhost:3000/edition');
      setEditions(response.data);
    } catch (error) {
      console.error('Error fetching editions:', error);
    }
  };

  fetchEditions();
}, []);
```

Puis on récupère les ouvrages de cette édition :

```
useEffect(() => {
  const fetchOuvrages = async () => {
    if (selectedEditionId) {
      try {
        const response = await axios.get(`http://localhost:3000/edition/${selectedEditionId}/ouvrages`);
        console.log(response.data);
        setOuvrages(response.data.ouvrages);
      } catch (error) {
        console.error('Error fetching ouvrages:', error);
      }
    }
  };

  fetchOuvrages();
}, [selectedEditionId]);
```

Puis enfin, on récupère les référents de cet établissement :

```
useEffect(() => {
  const fetchReferents = async () => {
    try {
      const token = document.cookie.replace(/(?:(?:^|.*;\s*)token\s*\=\s*([^;]*).*$)|^.*$/, "$1");
      const response = await axios.get(`http://localhost:3000/referents/${idEtablissement}`, {
          headers: {
              'Authorization': `Bearer ${token}`
          }
      });
      setReferents(response.data.referents);
    } catch (error) {
      console.error('Error fetching referents:', error);
    }
  };

  fetchReferents();
}, [idEtablissement]);
```

L'utilisateur entre les autres informations dans le formulaire de soumission d'un vœu. Enfin, on submit le formulaire :

```
const handleSubmit = async (event) => {
  event.preventDefault();
  try {
    const token = document.cookie.replace(/(?:(?:^|.*;\s*)token\s*\=\s*([^;]*).*$)|^.*$/, "$1");
    const response = await axios.post(`http://localhost:3000/edition/${selectedEditionId}/etablissement/${idEtablissement}/voeux`, {
      idOuvrage: selectedOuvrageId,
      idRef: selectedReferentId,
      prio,
    }, {
      headers: {
        'Authorization': `Bearer ${token}`
      }
    });
    console.log(response.data);
    setSuccessMessage('Voeu soumis avec succès !');
    setSelectedEditionId('');
    setSelectedOuvrageId('');
    setSelectedReferentId('');
    setPrio('');
    setError('');
    // Afficher un message de succès
  } catch (error) {
    console.error('Error submitting voeu:', error);
    setError('Erreur lors de la soumission du voeu');
  }
};
```

Le composant gère aussi l'affichage du formulaire :

```
return (
  <div className="max-w-md mx-auto mt-8">
    <h2 className="text-2xl font-bold mb-4">Soumettre un voeu</h2>
    <form onSubmit={handleSubmit} className="space-y-4">
      <div>
        <label htmlFor="edition" className="block font-semibold">Sélectionner une édition</label>
        <select id="edition" value={selectedEditionId} onChange={(e) => setSelectedEditionId(e.target.value)} className="block w-full py-2 px-3 border border-gray-300 rounded-m
          <option value="">Choisir une édition</option>
          {editions.map((edition) => (
            <option key={edition.idedition} value={edition.idedition}>{edition.description}</option>
          ))}
        </select>
      </div>
      {/* Champs pour sélectionner l'ouvrage */}
      <div>
        <label htmlFor="ouvrage" className="block font-semibold">Sélectionner un ouvrage</label>
        <select id="ouvrage" value={selectedOuvrageId} onChange={(e) => setSelectedOuvrageId(e.target.value)} className="block w-full py-2 px-3 border border-gray-300 rounded-m
          <option value="">Choisissez d'abord une édition</option>
          {ouvrages && ouvrages.map((ouvrages) => (
            <option key={ouvrages.idouvrage} value={ouvrages.idouvrage}>{ouvrages.titre}</option>
          ))}
        </select>
      </div>
      {/* Champs pour sélectionner le référent */}
      <div>
        <label htmlFor="referent" className="block font-semibold">Sélectionner un référent</label>
        <select id="referent" value={selectedReferentId} onChange={(e) => setSelectedReferentId(e.target.value)} className="block w-full py-2 px-3 border border-gray-300 rounde
          <option value="">Choisir un référent</option>
          {referents.map((referent) => (
            <option key={referent.idref} value={referent.idref}>{`${referent.nom} ${referent.prenom}`}</option>
          ))}
        </select>
      </div>
```

Voici une explication de son fonctionnement :

1. **Initialisation des états :**
    - Plusieurs états sont initialisés à l'aide du Hook useState de React pour gérer les données du formulaire, les messages d'erreur et de succès.
2. **Effets secondaires avec useEffect :**
    - Trois effets secondaires sont définis à l'aide du Hook useEffect. Chacun de ces effets déclenche une action lorsque certaines dépendances changent :
        - Le premier effet charge les éditions disponibles lors du chargement initial du composant.
        - Le deuxième effet charge les ouvrages disponibles lorsqu'une édition est sélectionnée.
        - Le troisième effet charge les référents disponibles pour l'établissement spécifié.
3. **Soumission du formulaire :**
    - Lorsque l'utilisateur soumet le formulaire, la fonction handleSubmit est appelée.
    - Cette fonction envoie une requête POST à l'API avec les données du formulaire, y compris l'identifiant de l'édition sélectionnée, l'identifiant de l'ouvrage choisi, l'identifiant du référent sélectionné et la priorité saisie.

- En cas de succès, un message de succès est affiché et les champs du formulaire sont réinitialisés.
- En cas d'erreur, un message d'erreur est affiché.

4. **Rendu du formulaire :**
   - Le formulaire comprend plusieurs champs :
     - Un sélecteur pour choisir une édition parmi celles disponibles.
     - Un autre sélecteur pour choisir un ouvrage associé à l'édition sélectionnée.
     - Un sélecteur pour choisir un référent parmi ceux disponibles pour l'établissement.
     - Un champ pour saisir la priorité du voeu.
     - Un bouton de soumission du formulaire.
     - Des éléments <p> sont utilisés pour afficher les messages de succès et d'erreur.

En résumé, ce composant permet à l'utilisateur de soumettre un voeu pour une édition spécifique en choisissant un ouvrage et un référent, puis en saisissant une priorité. Les données sont envoyées à l'API via une requête POST, et les messages de succès ou d'erreur sont affichés en fonction du résultat de la requête.

Modélisation des fonctionnalités

Malheureusement, nous n'avons pas eu le temps de mettre complétement la modélisation des fonctionnalités, il est donc possible qu'il y ait des incohérences entre ce qui est écrit ensuite et l'implantation dans le site web.

Inscription d'un établissement ou d'un auteur

Objectif : Permettre à un auteur ou un établissement de s'inscrire dans le système d'information.
Acteurs impliqués : Auteur (Principal), Etablissement (Principal)
Événement déclencheur : L'utilisateur souhaite s'inscrire au festival
Préconditions :
Enchaînement des actions :

- L'utilisateur clique sur le bouton "s'inscrire"
- Il renseigne son type (Auteur ou Établissement)
- Il renseigne ses informations (adresse mail, mot de passe, identifiant, nom, prénom, numéro de téléphone, adresse, localisation, langues)
- Le booleen "verifié" est automatiquement mis à "FAUX" et le booleen "veutParticiper" est automatiquement mis à "VRAI"
- Le système vérifie que l'utilisateur n'est pas déjà inscrit. S'il est déjà inscrit, l'inscription est annulée.
- Les informations sont enregistrées dans le système d'information

Actions alternatives :

- Si l'utilisateur est déjà inscrit (son mail ou son identifiant existent déjà dans le système d'information), on lui envoie un message d'erreur "Cet utilisateur existe déjà"
- Si l'utilisateur entre des informations incohérentes (comme un mail du mauvais format par exemple), on lui indique de réessayer avec un message d'erreur.

Postconditions : L'utilisateur est inscrit dans la table Auteur ou Etablissement selon l'information donnée

Connexion d'un établissement ou d'un auteur

Objectif : Permet à un établissement ou à un auteur de se connecter sur le site web et d'accéder aux informations et aux fonctionnalités qui lui sont propres
Acteurs impliqués : Établissement (Principal), Auteur (Principal)
Événement déclencheur : L'utilisateur souhaite se connecter
Préconditions : L'utilisateur est déjà inscrit dans le système d'information
Enchaînement des actions :

- L'utilisateur clique sur le bouton "se connecter"
- Il renseigne son mail et son mot de passe dans un formulaire et valide.
- Si le mail et le mot de passe rentrées correspondent à un utilisateur enregistré dans la base, alors l'accès est validé.
- L'utilisateur accède à son portail personnalisé en fonction de son type (établissement ou auteur).

Actions alternatives :

- Si le mail et/ou le mot de passe ne correspond à aucun utilisateur, on lui envoie un message d'erreur "Les informations rentrées sont erronés, veuillez réessayer"

Postconditions : L'utilisateur est connecté et le site est affiche les informations selon qui s'est connecté.

Consulter les ouvrages

Objectif : Permet aux utilisateurs d'avoir la liste des ouvrages proposés
Acteurs impliqués : Commission scolaire (Primaire), Gestionnaire du système (Secondaire), Etablissement (Secondaire), Auteur (Secondaire)
Événement déclencheur : L'utilisateur veut voir la liste des ouvrages proposés pour le festival.
Préconditions : L'utilisateur est connecté
Enchaînement des actions :

- L'utilisateur clique sur "Afficher les ouvrages"
- Le système vérifie quels ouvrages sont entrés dans la base et les envoie au site
- La liste des ouvrages s'affiche

Actions alternatives :

- Si il n'y a pas d'ouvrage, afficher "0 ouvrage trouvé dans le système d'information"

Postconditions :

Consulter les établissements

Objectif : Permet aux utilisateurs d'avoir la liste des établissements inscrits
Acteurs impliqués : Commission scolaire (Primaire), Gestionnaire du système (Secondaire)
Événement déclencheur : L'utilisateur veut voir la liste des établissements inscrits
Préconditions : L'utilisateur est connecté
Enchaînement des actions :

- L'utilisateur clique sur "afficher les établissements"
- Le système vérifie quels établissements sont inscrits et les envoie au site
- La liste des établissements s'affiche (toutes les informations sauf mot de passe)

Actions alternatives :

- Si il n'y a pas d'établissement, afficher "0 établissement trouvé dans le système d'information"

Postconditions :

Consulter les auteurs

Objectif : Permet aux utilisateurs d'avoir la liste des auteurs inscrits
Acteurs impliqués : Commission scolaire (Primaire), Gestionnaire du système (Secondaire)
Événement déclencheur : L'utilisateur souhaite consulter la liste des auteurs inscrits
Préconditions : L'utilisateur est connecté
Enchaînement des actions :

- L'utilisateur clique sur "Afficher les auteurs"
- Le système vérifie quels auteurs sont inscrits et les envoie au site
- La liste des auteurs s'affiche (toutes les informations sauf mot de passe)

Actions alternatives :

- Si il n'y a pas d'auteurs enregistrés, afficher "0 auteur trouvé dans le système d'information"

Postconditions :

Ajouter un ouvrage

Objectif : Permet à un auteur de proposer un de ses ouvrages à la commission
Acteurs impliqués : Auteur (Primaire)
Événement déclencheur : Un auteur souhaite ajouter un de ses ouvrages au festival
Préconditions : l'ouvrage n'est pas enregistré dans le système d'information et l'auteur est authentifié
Enchaînement des actions :

- Entrée des informations de l'ouvrage (nom, descriptions, langue…) par l'auteur
- Le système vérifie que l'ouvrage ne soit pas déjà dans la base de données
- L'ouvrage est ajouté dans le système d'information

Actions alternatives :

- Si l'ouvrage est déjà dans le système d'information alors :

Ne rien faire et envoyer un message d'erreur à l'auteur

Postconditions : L'ouvrage est bien enregistré dans la table Ouvrages

Rechercher un ouvrage

Objectif : Permet à un utilisateur de rechercher un ouvrage dans le système d'information
Acteurs impliqués : Etablissement (Secondaire), Auteur (Primaire), Commission scolaire (Secondaire)
Evénement déclencheur : L'utilisateur souhaite rechercher un ouvrage inscrit pour le festival
Préconditions : L'utilisateur est authentifié
Enchaînement des actions :

- Entrée du nom de l'ouvrage dans la barre de recherche
- Le système vérifie que l'ouvrage demandé soit bien inscrit dans la base de données
- Affichage des ouvrages ayant le nom entré dans la barre de recherche
- Affichage des informations (auteurs, langues...) pour chaque ouvrage

Actions alternatives :

- Si aucun résultat ne correspond, afficher "0 résultat correspondant"

Postconditions :

Modifier un ouvrage

Objectif : Permet à l'auteur de modifier les informations de son ouvrage
Acteurs impliqués : Auteur (Primaire)
Evénement déclencheur : L'auteur souhaite modifier les informations sur l'un de ses ouvrages
Préconditions : L'ouvrage qui va être modifié est bien enregistré dans le système d'information, l'auteur est authentifié
Enchaînement des actions :

- Selection de l'ouvrage à modifier par l'auteur
- Le système vérifie que l'auteur a bien écrit l'ouvrage recherché
- Ouverture du formulaire de modification des informations
- L'auteur entre les nouvelles informations (nom, description, langue…)
- Le système vérifie que les informations soient bien modifiées
- Enregistrement des informations dans le base de données

Actions alternatives :

- Si l'auteur n'a pas entré de modifications
  - ne rien faire

Postconditions : Les modifications ont bien été enregistrées dans la table Ouvrages

Supprimer un ouvrage

Objectif : Permet à un auteur de supprimer un ouvrage qu'il a préalablement proposé
Acteurs impliqués : Auteur (Primaire)
Evénement déclencheur : L'auteur souhaite supprimer un ouvrage qu'il a inscrit pour le festival
Préconditions : L'ouvrage qui va être supprimer est bien enregistré dans le système d'information et l'auteur est authentifié
Enchaînement des actions :

- Selection de l'ouvrage à supprimer par l'auteur
- Le système vérifie que l'auteur a bien écrit l'ouvrage sélectionné
- Suppression de l'ouvrage du système d'information
- On notifie l'auteur que l'ouvrage est bien supprimé avec un message

Actions alternatives :
Postconditions : L'ouvrage n'est plus dans la table Ouvrages

Rechercher un auteur

Objectif : Permet à un utilisateur de rechercher un auteur et ses informations
Acteurs impliqués : Etablissement (Primaire), Commission Scolaire(Secondaire)
Evénement déclencheur : L'utilisateur souhaite rechercher un auteur parmi les inscrits
Préconditions : L'acteur qui fait la recherche est authentifié
Enchaînement des actions :

- L'acteur qui fait la recherche entre le nom de l'auteur dans la barre de recherche
- Le système vérifie que l'auteur soit inscrit pour le festival et envoie ses informations au site
- Affichage des informations de l'auteur

Actions alternatives :

- Si aucun résultat ne correspond (l'auteur n'est pas dans le système d'information), afficher "0 résultat correspondant"

Postconditions :

Modifier un auteur

Objectif : Permet à un utilisateur de modifier les informations d'un auteur
Acteurs impliqués : Auteur (Primaire)
Événement déclencheur : L'auteur souhaite modifier ses informations
Préconditions : L'auteur est enregistré dans le système d'informations, l'auteur est authentifié
Enchaînement des actions :

- L’auteur clique sur son profil
- L’auteur modifie ses informations via les différents champs (nom, description, langue, ..)
- Le système vérifie que l’auteur a bien modifié ses informations
- Les informations sont enregistrées dans le système.

Actions alternatives :
- Si l’auteur n’a pas entré de modifications
    - ne rien faire

Postconditions : Les modification sont bien enregistrées dans la table Auteur

Supprimer un auteur

Objectif : Permet de supprimer un auteur du système d’information
Acteurs impliqués : Commission scolaire (Primaire), Auteur (Secondaire), Gestionnaire du système (Secondaire)
Événement déclencheur : L’utilisateur souhaite supprimer un auteur de la liste des inscrits
Préconditions : L'auteur à supprimer est enregistré dans le système, l'utilisateur est authentifié
Enchaînement des actions :
- Si la demande de suppression se fait par l’auteur
    - L’utilisateur clique sur “supprimer mon compte”
    - Les potentiels ouvrages enregistrés proposés par cet auteur sont supprimés de la table Ouvrages
    - L’auteur est supprimé de la table Auteurs
- Si la demande de suppression se fait par la commission scolaire ou le gestionnaire du système :
    - L’utilisateur accède à la liste des auteurs
    - L’utilisateur sélectionne l’auteur a supprimer
    - Le système vérifie que l’auteur est bien inscrit dans la table Auteurs
    - L’auteur est supprimé de la table Auteur
    - Les potentiels ouvrages enregistrés proposés par cet auteur sont supprimés de la table Ouvrages

Actions alternatives :
Postconditions : L’auteur supprimé n’est plus dans la table Auteurs et ses ouvrages ne sont plus dans la table Ouvrages

Rechercher un établissement

Objectif : Permet à la commission de rechercher des informations sur un établissement
Acteurs impliqués : Commission scolaire (Primaire)
Événement déclencheur : L’utilisateur souhaite rechercher un établissement inscrit au festival
Préconditions : L’utilisateur est authentifié
Enchaînement des actions :
- La commission entre le nom de l’établissement dans la barre de recherche
- Le système vérifie que l’établissement fait partie des établissements inscrits au festival
- Affichage de ou des établissements correspondants à la recherche
- Sélection de l’établissement dont on veut connaître les informations

- Le système recherche les informations sur l'établissement sélectionné et les envoie au site
- Affichage des informations de l'établissement sélectionné

Actions alternatives :

- Si aucun résultat ne correspond à la recherche, afficher "0 résultat correspondant"

Postconditions :

Modifier un établissement

Objectif : Permet à un utilisateur de modifier les informations d'un établissement
Acteurs impliqués : Etablissement (Primaire)
Evénement déclencheur : L'établissement souhaite modifier ses informations
Préconditions : L'établissement est enregistré dans le système d'information, l'établissement est authentifié
Enchaînement des actions :

- L'établissement clique sur son profil
- L'établissement modifie ses informations via les différents champs
- Le système envoie les modifications à la base de données
- Les modifications sont enregistrées

Actions alternatives :

- Si l'établissement n'a pas entré de modification
    - ne rien faire

Postconditions : Les modification sont bien enregistrées dans la table Etablissements

Supprimer un établissement

Objectif : Permet de supprimer un établissement du système d'information
Acteurs impliqués : Commission scolaire (Secondaire), Auteur (Primaire), Gestionnaire du système (Secondaire)
Événement déclencheur : L'utilisateur souhaite supprimer un auteur
Préconditions : L'établissement est dans la bdd, l'utilisateur est authentifié
Enchaînement des actions :

- Si la demande de suppression se fait par l'établissement
    - L'utilisateur clique sur "supprimer mon compte"
    - L'établissement est supprimé de la table Etablissements
- Si la demande de suppression se fait par la commission scolaire ou le gestionnaire du système :
    - L'utilisateur accède à la liste des établissements
    - L'utilisateur sélectionne l'établissement a supprimer
    - Le système vérifie que l'établissement est bien inscrit au festival)
    - L'établissement est supprimé de la table Etablissements

Actions alternatives :
Postconditions : L'établissement supprimé n'est plus dans la table Etablissements

Consultation des interventions

Objectif : Permet à l'utilisateur de voir toutes les interventions planifiées
Acteurs impliqués : Commission scolaire (Primaire), gestionnaire du système (Secondaire)
Événement déclencheur : L'utilisateur souhaite afficher les interventions
Préconditions : L'utilisateur est authentifié, la campagne des voeux est terminée
Enchaînement des actions :

- L’utilisateur recherche les interventions
- Le système recherche les intervention et les envoie au site
- Le site affiche les interventions

Actions alternatives :
Postconditions :

Consultation des interventions d’un établissement spécifique

Objectif : Permet à l'utilisateur de voir toutes les interventions planifiées pour son établissement
Acteurs impliqués : Etablissement (Primaire), Commission scolaire (Primaire)
Événement déclencheur : L’utilisateur souhaite consulter les interventions d’un établissement
Préconditions : L'utilisateur est authentifié, la campagne des voeux est terminée
Enchaînement des actions :

- Récupération de l’identifiant de l’établissement
- Recherche des interventions de cet établissement par le système
- Affichage des interventions correspondantes

Actions alternatives :
Postconditions :

Consultation des voeux

Objectif : Permet à l'utilisateur d’obtenir la liste des voeux
Acteurs impliqués : Commission scolaire (Primaire), gestionnaire du système (Secondaire)
Événement déclencheur : L’utilisateur souhaite consulter les voeux
Préconditions : L’utilisateur est authentifié, l’édition du festival est en cours
Enchaînement des actions :

- L’utilisateur clique sur rechercher les vœux
- Le système récupère tous les vœux inscrits dans la base de données et les envoie au site
- Affichage de tous les voeux pour l’édition actuelle

Actions alternatives : Si aucun voeu n’a été formulé alors afficher “Pas de voeu formulé”, si l’édition du festival est terminé afficher “Le festival n’est pas en cours”
Postconditions :

Consultation des voeux d’un établissement spécifique

Objectif : Permet à l'utilisateur de voir les voeux qu’il a émis
Acteurs impliqués : Etablissement (Primaire), Commission scolaire (Praimaire)
Événement déclencheur : L’utilisateur souhaite rechercher les vœux d’un établissement
Préconditions : L’acteur est authentifié, l’édition du festival est en cours
Enchaînement des actions :

- L’utilisateur recherche un établissement
- Le système vérifie que l’établissement est inscrit et envoie les vœux au site
- Affichage de tous les voeux réalisés pour l’établissement concerné

Actions alternatives : Si aucun voeu n'a été formulé alors afficher "Pas de voeu formulé", si l'édition du festival est terminé afficher "Le festival n'est pas en cours"
Postconditions :

Notifier sa volonté de participer

Objectif : Permet à un auteur inscrit de notifier sa volonté de participer à la prochaine édition via l'application web
Acteurs impliqués : Auteur (Principal), Etablissement (Secondaire)
Événement déclencheur : Un auteur ou un établissement souhaite participer au festival
Préconditions : Acteur authentifié, la colonne de l'auteur veutParticiper ==FALSE
Enchaînement des actions :

- L'utilisateur rentre ses informations dans le système
- Le système vérifie que l'acteur n'est pas déjà inscrit au festival
- Modifier la volonté de participer (veutParticiper –> TRUE) de l'auteur ou l'établissement authentifié et l'enregistrer dans la base de données

Actions alternatives :
Postconditions : la modification a bien été enregistrées dans la table Auteur ou Etablissement en fonction des informations rentrées

Générer les statistiques sur les ouvrages

Objectif : Générer les statistiques sur les ouvrages automatiquement à la fin des inscriptions pour chaque édition du festival
Acteurs impliqués : Automatique
Événement déclencheur : à la clôture des inscriptions (= le jour suivant de finInscriptions)
Préconditions :
Enchaînement des actions :

- Compter le nombre d'ouvrages sélectionnés pour l'édition
- Compter le nombre d'ouvrages ayant pour public cible "jeune enfant", "enfant" ou "ado"
- Compter le nombre d'ouvrages ayant pour public cible "jeune adulte" ou "adulte"
- Enregistrer ces statistiques dans la base de données pour l'édition en cours (même année que l'édition en cours)

Actions alternatives :
Postconditions : les statistiques ont bien été enregistrées dans la table StatsOuvrages

Consulter les statistiques sur les ouvrages

Objectif : Permet à tout le monde de pouvoir consulter les statistique sur les ouvrages sélectionnés de l'édition voulu
Acteurs impliqués : Tous (Principal)
Événement déclencheur : L'utilisateur souhaite consulter les statistiques des oubrages
Préconditions : les inscriptions sont terminées pour l'édition choisie
Enchaînement des actions :

- L'utilisateur choisi l'édition pour laquelle il veut les statistiques
- Le système vérifie qu'une édition ait eu lieu cette année et envoie les informations au site

- Affichage des statistiques sur les ouvrages de l'année choisie

Actions alternatives :
Postconditions :

Générer les statistiques sur la campagne de voeux

Objectif : Générer les statistiques sur la campagne de voeux automatiquement à la fin de la campagne de voeux pour chaque édition du festival
Acteurs impliqués : Automatique
Événement déclencheur : à la clôture de la campagne de voeux (= le jour suivant de finVoeux)
Préconditions :
Enchaînement des actions :

- Compter le nombre d'établissements inscrits à l'édition en cours
- Compter le nombre d'établissements ayant soumis au moins un voeux pour l'édition en cours
- Calculer le taux de participation des établissement à la campagne de voeux (=nbParticipants / nb ayant soumis des voeux)
- Compter le nombre de voeux soumis pour l'édition en cours
- Calculer le nombre moyen de voeux soumis par les établissements inscrits (=nbVoeux / nbParticipants)
- Enregistrer ces statistiques dans la base de données pour l'édition en cours (même année que l'édition en cours)

Actions alternatives :
Postconditions : les statistiques ont bien été enregistrées dans la table StatsCampagneVoeux

Consulter les statistiques sur la campagne de voeux

Objectif : Permet à tout le monde de pouvoir consulter les statistique sur la campagne de voeux de l'édition voulu
Acteurs impliqués : Tous (Principal)
Événement déclencheur : L'utilisateur souhaite consulter les statistiques de la campagne des voeux
Préconditions : la campagne de voeux est terminée pour l'édition choisie
Enchaînement des actions :

- L'utilisateur choisi l'édition pour laquelle il veut les statistiques
- Le système vérifie qu'une édition ait eu lieu cette année et envoie les informations au site
- Affichage des statistiques sur la campagne de voeux de l'année choisie

Actions alternatives :
Postconditions :

Générer les statistiques sur les interventions

Objectif : Générer les statistiques sur les interventions automatiquement à la fin de du festival pour chaque édition du festival
Acteurs impliqués : Automatique
Événement déclencheur : à la clôture du festival (= le jour suivant de finFestival)
Préconditions :
Enchaînement des actions :

- Compter le nombre d'intervention réalisées (= toutes les intervention d'état "planifié")

- Compter le nombre total d'élèves présents à ces interventions (/!\nbEleves peut être Null)
- Calculer le nombre moyen d'élèves à ces intervention (=nbElevesCumulé / nombre d'intervention "planifiée" avec nbEleves Not Null)
- Enregistrer ces statistiques dans la base de données pour l'édition en cours (même année que l'édition en cours)

Actions alternatives :
Postconditions : les statistiques ont bien été enregistrées dans la table StatsInterventions

Consulter les statistiques sur les interventions

Objectif : Permet à tout le monde de pouvoir consulter les statistiques sur les interventions de l'édition voulu
Acteurs impliqués : Tous (Principal)
Événement déclencheur : L'utilisateur souhaite consulter les statistiques sur les intervention d'une édition spécifique
Préconditions : le festival est terminé pour l'édition choisie
Enchaînement des actions :

- L'utilisateur choisi l'édition pour laquelle il veut les statistiques
- Le système vérifie qu'une édition ait eu lieu cette année et envoie les informations au site
- Affichage des statistiques sur les interventions de l'année choisie

Actions alternatives :
Postconditions :

Création d'une édition

Objectif : planifier une nouvelle édition du festival avec sa description et ses dates
Acteurs impliqués : Commission Scolaire (Principal)
Événement déclencheur : La commission scolaire souhaite crée une édition du festival
Préconditions :
Enchaînement des actions :

- la commission scolaire renseigne les dates clés de l'édition et rempli la description
- Le système vérifie qu'une édition aux mêmes dates ne soit pas déjà inscrite
- enregistrement d'une nouvelle édition dans la base de données avec les informations saisies

Actions alternatives :
Postconditions : une nouvelle édition a été enregistrée dans la table Edition

Modification d'une édition

Objectif : permettre à la commission scolaire d'apporter des modification sur une édition enregistrée à venir
Acteurs impliqués : Commission Scolaire (Principal), Tous (Secondaire)
Événement déclencheur : La commission scolaire souhaite modifier les informations d'une édition
Préconditions : l'édition à modifier ne doit pas avoir fini ses inscriptions
Enchaînement des actions :

- la commission scolaire renseigne les dates ou la description à modifier (/!\on ne peut pas modifier une date déjà passée)

- Le système vérifie que les dates entrées correspondent a l'édition à venir
- prévenir par mail de la modification les acteurs concernés par l'édition

Actions alternatives :
Postconditions : les modifications ont bien été enregistrées dans la table Edition

Sélection d'ouvrages

Objectif : permettre à la commission scolaire de choisir parmis les ouvrages enregistrés par les auteurs souhaitant participer, ceux qui seront sélectionnés pour la nouvelle édition du festival
Acteurs impliqués : Commission Scolaire (Principal), Auteur (Secondaire)
Événement déclencheur : La commission scolaire souhaite choisir les ouvrages pour l'édition à venir
Préconditions : le festival concerné n'a pas encore commencé sa campagne de voeux
Enchaînement des actions :

- L'utilisateur demande à afficher les oubrages proposés
- afficher la liste des ouvrages proposés par les auteurs voulant participer
- la commission scolaire sélectionne ou désélectionne les ouvrages qui seront sélectionnés pour le festival en cours
- Le système enregistre les sélections et les envoie à la base de données
- tous les ouvrages sélectionnés sont enregistrés comme liés à l'édition concernée dans la base de données

Actions alternatives :
Postconditions : Les informations sont enregistrées dans la table Ouvrage

Sélection d'établissements

Objectif : permettre à la commission scolaire de choisir parmis les établissements souhaitant participer, ceux qui seront sélectionnés pour la nouvelle édition du festival
Acteurs impliqués : Commission Scolaire (Principal), Etablissement (Secondaire)
Événement déclencheur : La commission scolaire souhaite sélectionner les établissements pour l'édition à venir
Préconditions : le festival concerné n'a pas encore commencé sa campagne de voeux
Enchaînement des actions :

- L'utilisateur demande à afficher la liste des établissement souhaitant participer
- afficher la liste des établissements voulant participer
- la commission scolaire sélectionne ou désélectionne les établissements qui seront sélectionnés pour le festival en cours
- Le système enregistre les sélections et les envoie à la base de données
- tous les établissements sélectionnés sont enregistrés comme liés à l'édition concernée dans la base de données

Actions alternatives :
Postconditions : Les informations sont enregistrées dans la table Etablissement

Fin des inscriptions : on prévient les auteurs et établissements participants

Objectif : prévenir les auteurs et les établissements qui ont étés sélectionnés pour le festival à la fin des inscriptions

Acteurs impliqués : Automatique (Principal), Auteur (Secondaire), Etablissement (Secondaire)
Événement déclencheur : fin des inscriptions (jour suivant finInscriptions)
Préconditions :
Enchaînement des actions :

- Vérifier dans la base de donnée les auteurs et établissements sélectionnés
- Prévenir par mail les les établissements sélectionnés
- Prévenir par mail les auteurs ayant un ouvrage sélectionné

Actions alternatives :
Postconditions :

Fin de la campagne de voeux : récapitulatif des interventions à venir et refus automatique des voeux

Objectif : envoie un récapitulatif aux auteurs et établissements sélectionnés pour le festival, des interventions qui leurs ont été attribuées
Acteurs impliqués : Automatique (Principal), Auteur (Secondaire), Etablissement (Secondaire)
Événement déclencheur : fin de la campagne de voeux (jour suivant finInscriptions)
Préconditions :
Enchaînement des actions :

- Le système recherche les vœux dans la base de données
- Pour chaque auteur ou établissement sélectionné envoyer par mail un récapitulatif de ses interventions à venir (dates, acteurs concernés…)
- Tous les voeux qui ne sont pas validés passent à l'état "refusé"

Actions alternatives :
Postconditions :

Fin de l'édition (Automatique) :

Objectif : permettre au système de réinitialiser, à la fin d'une édition, certaines données afin d'assurer son bon fonctionnement pour les prochaines éditions
Acteurs impliqués : Automatique (Principal)
Événement déclencheur : fin de l'édition du festival (jour suivant finFestival)
Préconditions :
Enchaînement des actions :

- Pour chaque auteur ou établissement réinitialise sa volonté de participer (veutParticiper -> FALSE)

Actions alternatives :
Postconditions : veutParticiper == FALSE pour tous les établissements ou auteur dans les tables Auteur et Etablissement

Enregistrer un accompagnateur ou un interprète

Objectif : renseigner et enregistrer des informations sur un accompagnateur ou un Interprète pour pouvoir le lier à une intervention
Acteurs impliqués : Commission Scolaire (Principal), Accompagnateur (Secondaire), Interprète (Secondaire)
Événement déclencheur :

- appel de la fonctionnalité à la planification d'une intervention ou d'un remplacement
- OU juste sur demande de la commission en prévention de futur interventions

Préconditions : L'Accompagnateur (ou l'Interprète) n'existe pas dans le système
Enchaînement des actions :

- la commission scolaire choisit s'il veut enregistrer un accompagnateur ou un interprète
- renseignement des informations sur l'accompagnateur ou l'interprète par la commission scolaire (nom, prénom, téléphone, mail + langues si interprète)
- Le système vérifie si l'accompagnateur ou l'interprète existe dans la base de données ou non
- enregistrement du nouvel accompagnateur ou interprète dans la base de données

Actions alternatives : si l'accompagnateur ou l'interprète existe déjà le notifier et ne pas enregistrer dans la base de donnée
Postconditions : l'accompagnateur ou l'interprète existe bien dans la table Interprete ou Accompagnateur

Émettre un Vœu

Objectif : Permet à un Etablissement d'émettre un voeu
Acteurs impliqués : Etablissement (Primaire), Référent (Secondaire)
Evénement déclencheur : Un etablissement souhaite émettre un vœu
Préconditions : la campagne de voeux est en cours, il existe une liste d'ouvrage à sélectionner, l'établissement est authentifié
Enchaînement des actions :

- L'établissement consulte la liste des ouvrage sélectionnés et sélectionne celui voulu
- L'établissement sélectionne une priorité (entre 1 et 3) parmi celles restantes (un établissement ne peut pas appliquer la même priorité sur 2 vœux)
- L'établissement définit un Référent :
  - s'il existe déjà il le sélectionne dans un menu déroulant où tous les référents figurent
  - sinon le référent est créé selon les informations renseignées par l'établissement et ajouté à la base de données
- Création du nouveau Vœu dans la base de données avec pour état "En attente", pour priorité celle choisi plus tôt et le lier au Référent choisi
- Prévenir par mail le référent concerné de l'émission du vœu

Actions alternatives :
Postconditions : Le vœu est bien enregistré dans la table Vœu

Modifier un vœu

Objectif : Permettre aux établissements de modifier les vœux déjà soumis (modifier la priorité, changer le référent, l'ouvrage…)
Acteurs impliqués : Établissement (Principal), Référent (Secondaire)
Événement déclencheur : L'établissement souhaite modifier un de ses voeux
Préconditions : Des vœux ont déjà été émis par l'établissement à cette édition, on est encore dans la période de campagne de vœux
Enchaînement des actions :

- L'établissement sélectionne via l'application le vœu à modifier
- Modifier le ou le champ voulu : les modifications doivent respecter les mêmes contraintes que lors de l'émission d'un vœu (priorité, référent…)
- Si les contraintes sont respectées : enregistrer les modifications dans la base de données

        - Sinon affiche le message d'erreur "Les contraintes n'ont pas été respectées"
    - Prévenir par mail le référent concerné de la modification du vœu

Actions alternatives :
Postconditions : Les modification ont bien été enregistrées dans la table Voeu

Supprimer un vœu :

Objectif : Permettre aux établissements de supprimer un des vœux qu'ils ont soumis pour l'édition en cours
Acteurs impliqués : Etablissement (principal)
Événement déclencheur : L'établissement souhaite supprimer un vœu
Préconditions : Des vœux ont déjà été émis par l'établissement à cette édition, on est encore dans la campagne de vœux
Enchaînement des actions :

- Sélection via l'application du vœu à supprimer par l'établissement
- Le système vérifie si le vœu st bien lié a l'établissement
- Supprimer le vœu du système d'information
- Prévenir par mail le référent concerné de la suppression du vœu

Actions alternatives :
Postconditions : Le vœu est supprimé de la table Vœu

Planifier une intervention

Objectif : La commission scolaire planifie une Intervention d'auteur dans un établissement selon les vœux soumis par les établissements à la fin de la campagne de vœux
Acteurs impliqués : Commission Scolaire (Principal), Auteur (Secondaire), Etablissement (Secondaire), Accompagnateur (Secondaire), Interprète (Secondaire), Réfèrent (Secondaire)
Événement déclencheur : La commission scolaire souhaite planifier une intervention
Préconditions :

- La campagne de vœux est terminée pour cette édition
- Il existe un vœu correspondant à l'intervention qui va être créée (même établissement, l'auteur de l'intervention est l'auteur de l'ouvrage sélectionner dans le vœu)

Enchaînement des actions :

- La commission scolaire rempli un formulaire pour renseigner la date et les heures de début et de fin ; sélectionne l'auteur, l'établissement et l'accompagnateur concerné et si besoin l'interprète
- Enregistrer dans la base de données une nouvelle intervention avec les informations renseignées
- Lier l'intervention avec l'auteur, l'établissement et l'accompagnateur choisi et, s'il y en a, l'interprète correspondant
- Prévenir par mail l'auteur, l'établissement, l'accompagnateur et l'interprète (si il y en a un)

Actions alternatives :

- Si l'accompagnateur ou l'interprète n'existent pas appeler les fonctionnalités de création d'accompagnateur ou d'interprète
- Si les informations renseignées ne sont pas correctes, afficher un message d'erreur et proposer de recommencer l'opération.

Postconditions : Les modifications ont bien été enregistrées dans la table Intervention

Annuler une intervention

Objectif : Permettre aux auteurs ou aux établissements de se désister et donc d'annuler une de leurs interventions planifiées
Acteurs impliqués : Etablissement (Principal), Auteur (Principal), Commission Scolaire (Secondaire), Accompagnateur (Secondaire), Interprète (Secondaire)
Evénement déclencheur : Un acteur souhaite annuler une intervention
Préconditions :

- L'établissement ou l'auteur concerné a une ou des interventions planifiées pour l'édition en cours
- La date de l'intervention concernée n'est pas encore passée

Enchaînement des actions :

- L'auteur ou l'établissement consulte la liste des ses interventions planifiées et choisi celle à annuler et renseigne s'il aimerait un remplacement ou non
- L'état de l'intervention passe de "planifiée" à "annulée" (modification dans la base de données)
- Un message de confirmation d'annulation est envoyé à l'établissement ou l'auteur qui vient de faire l'annulation
- Prévenir de l'annulation (et de la volonté ou non de remplacement) par mail l'auteur, l'établissement, la commission scolaire, l'accompagnateur et l'interprète (s'il y en a un) concernés par l'intervention

Actions alternatives :
Postconditions : Les modifications ont bien été enregistrées dans la table Intervention

Remplacer une intervention

Objectif : Permettre à la commission scolaire de planifier une nouvelle intervention s'il y a eu une annulation (on garde le même établissement et le même auteur)
Acteurs impliqués : Commission Scolaire (Principal), Etablissement (Secondaire), Auteur (Secondaire), Accompagnateur (Secondaire), Interprète (Secondaire)
Événement déclencheur :

- Une annulation d'intervention a été faite et une volonté de remplacement a été émise
- La commission scolaire décide de remplacer l'intervention

Préconditions :

- Une intervention à l'état "annulée" existe et concerne l'édition en cours
- On est après la campagne de vœux et avant la fin du festival

Enchaînement des actions :

- La commission scolaire sélectionne l'intervention annulée à remplacer
- La commission renseigne la nouvelle date et les nouveaux horaires
- La commission renseigne, si besoin, le nouvel accompagnateur et le nouvel interprète
- Créer une nouvelle intervention concernant le même auteur et le même établissement que l'intervention annulée avec pour état "planifiée", avec la nouvelle date et les nouveaux horaires, avec le nouvel accompagnateur et le nouvel interprète (si besoin)
- L'intervention annulée passe à l'état "remplacée"
- Enregistrer dans la base de données la création de la nouvelle intervention et la modification de l'ancienne
- Prévenir par mail l'auteur, l'établissement, l'accompagnateur et l'interprète (s'il y en a un) de la nouvelle intervention programmée

Actions alternatives :
Postconditions : Les modifications ont bien été enregistrées dans la table Intervention

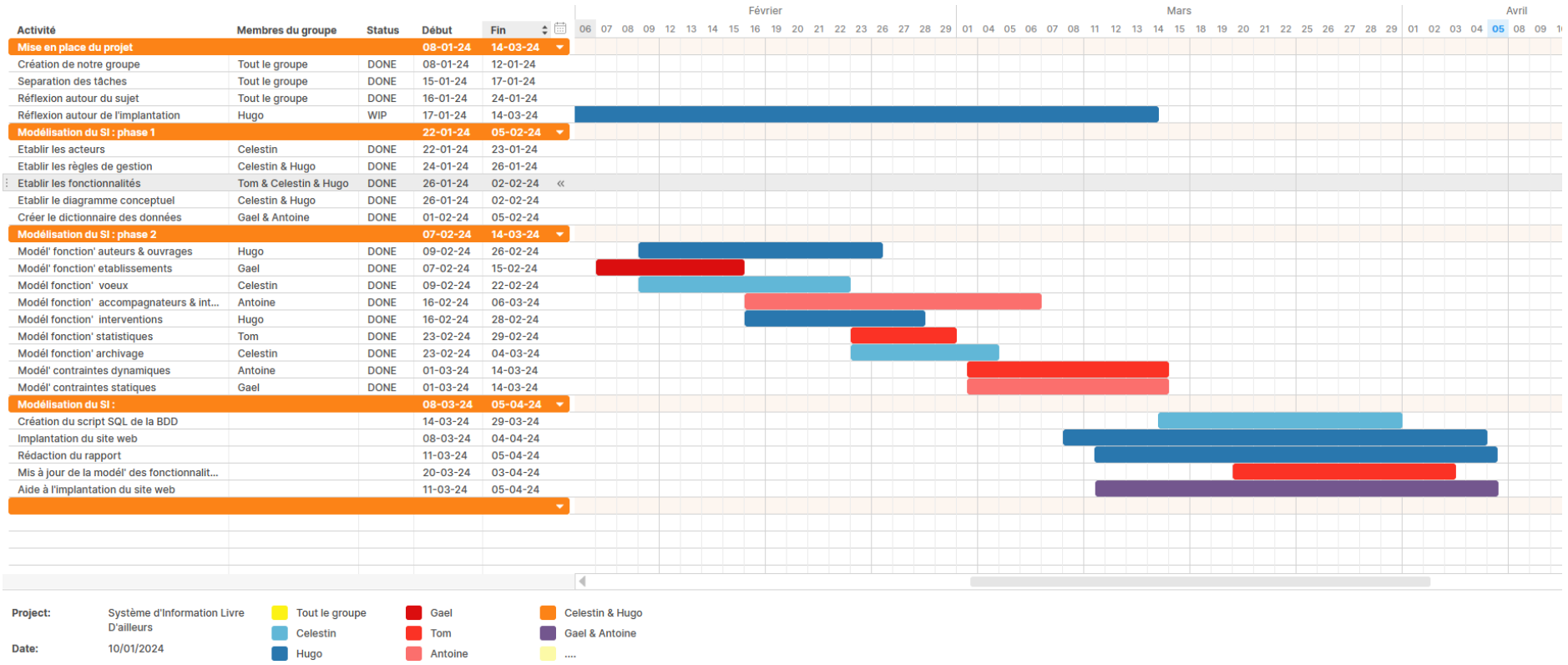
Renseigner le nombre d'élèves

Objectif : permettre à la commission scolaire de renseigner le nombre d’élèves présents à une intervention
Acteurs impliqués : Commission Scolaire (Principal)
Événement déclencheur : après une intervention, la commission scolaire veut renseigner le nombre d’élèves
Préconditions : l’intervention concernée est passée
Enchaînement des actions :

- la commission scolaire choisi une intervention parmis la liste des intervention planifiées de l’édition en cours
- renseigne le nombre d’élève présent dans le champs correspondant et valide
- la modification dans l’intervention est enregistrée dans la base de données

Actions alternatives : Si l’intervention n’a pas encore eu lieu affiche un message d’erreur
Postconditions : la modification a bien été enregistrée dans la table Intervention

Repartition du travail au sein du groupe



Activité	Membres du groupe	Status	Début	Fin
Mise en place du projet			**08-01-24**	**14-03-24**
Création de notre groupe	Tout le groupe	DONE	08-01-24	12-01-24
Separation des tâches	Tout le groupe	DONE	15-01-24	17-01-24
Réflexion autour du sujet	Tout le groupe	DONE	16-01-24	24-01-24
Réflexion autour de l'implantation	Hugo	WIP	17-01-24	14-03-24
Modélisation du SI : phase 1			**22-01-24**	**05-02-24**
Etablir les acteurs	Celestin	DONE	22-01-24	23-01-24
Etablir les règles de gestion	Celestin & Hugo	DONE	24-01-24	26-01-24
Etablir les fonctionnalités	Tom & Celestin & Hugo	DONE	26-01-24	02-02-24
Etablir le diagramme conceptuel	Celestin & Hugo	DONE	26-01-24	02-02-24
Créer le dictionnaire des données	Gael & Antoine	DONE	01-02-24	05-02-24
Modélisation du SI : phase 2			**07-02-24**	**14-03-24**
Modél' fonction' auteurs & ouvrages	Hugo	DONE	09-02-24	26-02-24
Modél' fonction' etablissements	Gael	DONE	07-02-24	15-02-24
Modél fonction' voeux	Celestin	DONE	09-02-24	22-02-24
Modél fonction' accompagnateurs & int...	Antoine	DONE	16-02-24	06-03-24
Modél fonction' interventions	Hugo	DONE	16-02-24	28-02-24
Modél fonction' statistiques	Tom	DONE	23-02-24	29-02-24
Modél fonction' archivage	Celestin	DONE	23-02-24	04-03-24
Modél' contraintes dynamiques	Antoine	DONE	01-03-24	14-03-24
Modél' contraintes statiques	Gael	DONE	01-03-24	14-03-24
Modélisation du SI :			**08-03-24**	**05-04-24**
Création du script SQL de la BDD			14-03-24	29-03-24
Implantation du site web			08-03-24	04-04-24
Rédaction du rapport			11-03-24	05-04-24
Mis à jour de la modél' des fonctionnalit...			20-03-24	03-04-24
Aide à l'implantation du site web			11-03-24	05-04-24

Conclusion

Ce rapport présente l'étape finale du projet de conception de système d'informations. Il présente les règles de gestion, les acteurs, les diagrammes de classes et le dictionnaire des données. Bien sur, il inclut la modélisation des contraintes d'intégrité. Aussi, il présente l'implantation du SI sous forme de site web avec les framework utilisés.

Plus largement, ce projet nous a permis de developper des vraies compétences professionnelles sur la conception de SI ainsi que l'implantation dans une application web. Ce fut très challengeant étant donné la contrainte de temps et parfois le manque de formation sur certains points. Ce projet a donc été une superbe opportunité pour apprendre plus largement la CSI et la conception d'interface web moderne avec le stack PERN.