Documentation technique

ORGANIZHEUR

Sommaire

A. *Cahier des charges*

Contexte

Organiz'heur est une entreprise de prestation de services qui souhaite mieux organiser ses activités en créant des listes de tâches à affecter à ses collaborateurs avec possibilité de catégorisation. Une application est donc demandée avec les fonctionnalités suivantes :

Fonctionnalités principales

Gestion des utilisateurs

- Connexion via email et mot de passe
- Utilisateurs standards et administrateurs
- Association d'un utilisateur à une seule catégorie
- Possibilité de désactiver l'accès d'un utilisateur

Gestion des tâches

- Tâche : libellé, échéance (date+heure), état de complétion
- Traçabilité complète : créateur, modificateur, completeur avec timestamps
- Rattachement obligatoire à une liste

Gestion des listes

- Listes personnelles (non catégorisables) ou catégorisées
- Traçabilité : créateur et modificateur avec timestamps
- **Archivage automatique** quand toutes les tâches sont complétées

Gestion des catégories

- Créées/modifiées/supprimées uniquement par l'administrateur
- Association des listes aux catégories
- Gestion des droits d'accès par catégorie

Droits d'accès

- **Utilisateur standard** : accès aux listes de ses catégories + listes personnelles
- **Administrateur** :
    - Gestion complète des utilisateurs et catégories

- Vue récapitulative avec statistiques (tâches accomplies, retards)
- Analyse par utilisateur et par catégorie

Écrans prévus

1. Gestion de la connexion
2. Gestion des utilisateurs (admin uniquement)
3. Gestion des catégories (admin uniquement)
4. Gestion des listes
5. Gestion des tâches d'une liste

B. Dictionnaire de données

Afin de développer l'application, les données suivantes ont été retenues et confinées dans le dictionnaire de données suivant :

Donnée	Type	Description
IDCategorie	INT	Identifiant unique de la catégorie
libelleCategorie	VARCHAR (100)	Nom/libellé de la catégorie
mailEmploye	VARCHAR (50)	Adresse email de l'employé (identifiant unique)
MDPEmploye	VARCHAR (75)	Mot de passe hashé de l'employé
prenomEmploye	VARCHAR (50)	Prénom de l'employé
nomEmploye	VARCHAR (50)	Nom de l'employé
estAdmin	BOOLEAN	Indique si l'employé est administrateur
IDListe	INT	Identifiant unique de la liste
libelleListe	VARCHAR (255)	Nom/libellé de la liste
dateCreationListe	DATETIME	Date et heure de la création
dateMAJliste	DATETIME	Date et heure de la dernière modification
dateArchivageListe	DATETIME	Date et heure de l'archivage
Estpersonnelle	BOOLEAN	Indique si la liste est personnelle

estArchivee	BOOLEAN	Indique si la liste est archivée
IDTache	INT	Identifiant unique de la tache
libelleTache	VARCHAR (255)	Nom/libellé de la tâche
dateEcheanceTache	DATETIME	Date et heure d'échéance de la tâche
etatTache	BOOLEAN	Etat de complétion de la tâche
dateCreationTache	DATETIME	Date et heure de la création de la tâche
dateMAJTache	DATETIME	Date et heure de la dernière modification de la tâche
dateCompletionTache	DATETIME	Date et heure de la complétion de la tâche
mailMAJEmploye	VARCHAR (50)	Identifiant de l'employé ayant effectué la modification

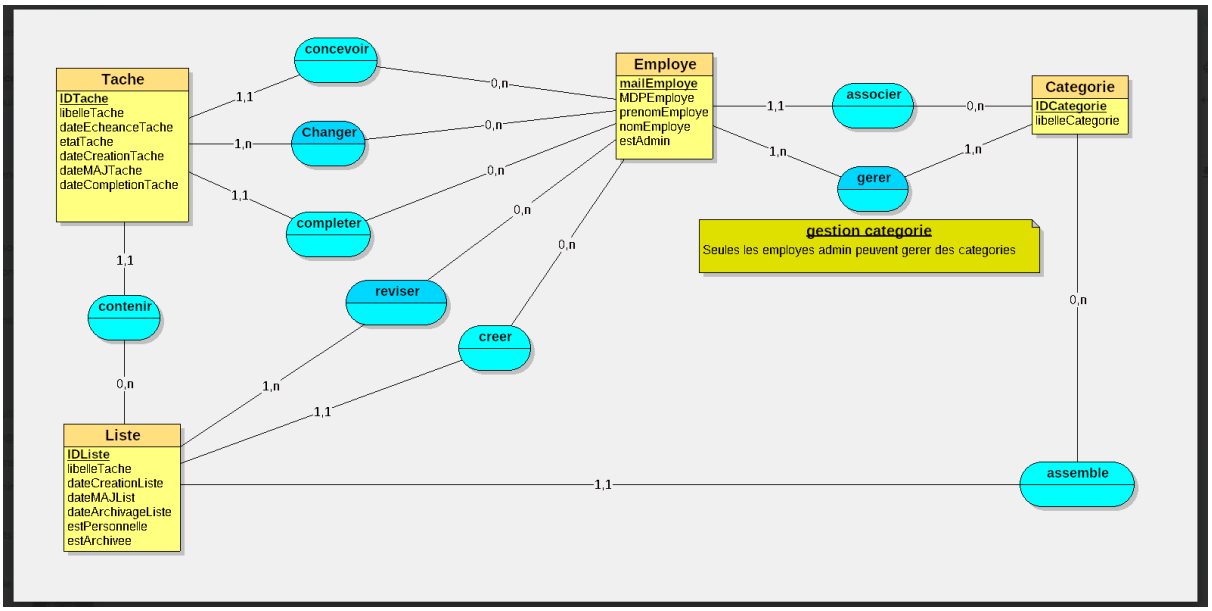
C. <u>Dépendances fonctionnelles</u>

Ces données permettent d'établir les liens suivants entre les différentes tables de la base de donnée :

- **mailEmploye** ? MDPEmploye, prenomEmploye, nomEmploye, estAdmin, IDCategorie
- **IDTache** ? libelleTache, dateEcheanceTache, etatTache, dateCreationTache, dateMAJTache, mailEmploye, mailMAJEmploye, IDListe
- **IDListe ?** libelleTache, dateCreationListe, dateMAJList, dateArchivageListe, estPersonnelle, estArchivee, mailEmploye, IDCategorie
- **IDCategorie ?** libelleCategorie
- **(mailEmploye, IDTache)** → dateCompletionTache *(table completer)*

D. MCD

Les relations entre les différentes tables de la base de donnée sont représentées pqr le schéma suivant :



E. MLD

Le schéma ci-dessus peut également être exprimée sous forme de relations logiques de la manière suivante :

- Employe = (mailEmploye VARCHAR(50), MDPEmploye VARCHAR(75), prenomEmploye VARCHAR(50), nomEmploye VARCHAR(60), estAdmin BOOLEAN, #IDCategorie);

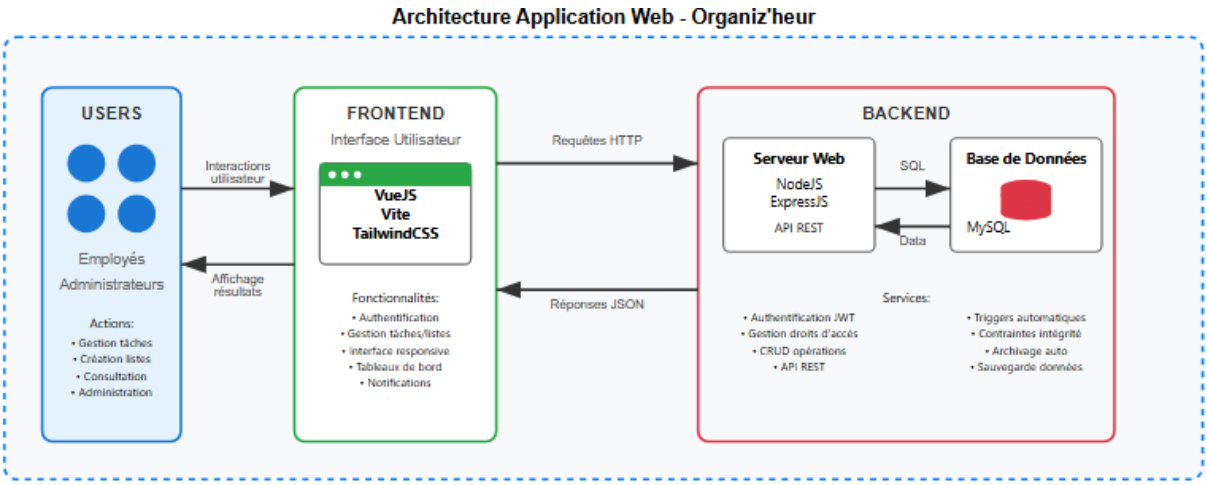
- Categorie = (IDCategorie INT, libelleCategorie VARCHAR(100));

- Liste = (IDListe INT, libelleTache VARCHAR(255), dateCreationListe DATETIME, dateMAJList DATETIME, dateArchivageListe DATETIME, estPersonnelle BOOLEAN, estArchivee BOOLEAN, #mailEmploye, #IDCategorie*);

- Tache = (IDTache INT, libelleTache VARCHAR(255), dateEcheanceTache DATETIME, etatTache BOOLEAN, dateCreationTache DATETIME, dateMAJTache DATETIME, #mailEmploye, #mailEmploye_1, #IDListe);

- reviser = (#mailEmploye, #IDListe);
- changer = (#mailEmploye, #IDTache);
- gerer = (#mailEmploye, #IDCategorie);
- completer = (#mailEmploye, #IDTache, dateCompletionTache DATETIME);

F. Schéma d'architecture technique



G. Script SQL

```
-- Table Categorie
CREATE TABLE Categorie(
IDCategorie INT AUTO-INCREMENT,
libelleCategorie VARCHAR(100) NOT NULL,
PRIMARY KEY(IDCategorie)
);

 -- Table Employe
CREATE TABLE Employe(
mailEmploye VARCHAR(50),
MDPEmploye VARCHAR(75),
prenomEmploye VARCHAR(50),
nomEmploye VARCHAR(50),
```

```
estAdmin BOOLEAN,
IDCategorie INT NOT NULL,
PRIMARY KEY(mailEmploye),
FOREIGN KEY(IDCategorie) REFERENCES Categorie(IDCategorie)
);

-- Table Liste avec contrainte CHECK pour listes personnelles
CREATE TABLE Liste(
IDListe INT AUTO-INCREMENT,
libelleListe VARCHAR(255),
dateCreationListe DATETIME,
dateMAJListe DATETIME, dateArchivageListe DATETIME,
estPersonnelle BOOLEAN,
estArchivee BOOLEAN,
mailEmploye VARCHAR(50) NOT NULL,
IDCategorie INT,

-- Suppression du NOT NULL pour permettre NULL pour les listes personnelles
PRIMARY KEY(IDListe),
FOREIGN KEY(mailEmploye) REFERENCES Employe(mailEmploye),
FOREIGN KEY(IDCategorie) REFERENCES Categorie(IDCategorie),

-- Ajout de la contrainte CHECK
CONSTRAINT chk_liste_personnelle CHECK (
(estPersonnelle = TRUE AND IDCategorie IS NULL) OR
(estPersonnelle = FALSE AND IDCategorie IS NOT NULL)
)
)

 -- Table Tache CREATE TABLE Tache(
IDTache INT AUTO-INCREMENT,
libelleTache VARCHAR(255),
dateEcheanceTache DATETIME,
etatTache BOOLEAN,
dateCreationTache DATETIME,
dateMAJTache DATETIME,
dateCompletionTache DATETIME,
mailEmploye VARCHAR(50) NOT NULL,
mailMAJEmploye VARCHAR(50) NOT NULL,
IDListe INT NOT NULL, PRIMARY KEY(IDTache),
FOREIGN KEY(mailEmploye) REFERENCES Employe(mailEmploye),
FOREIGN KEY(mailEmploye_1) REFERENCES Employe(mailEmploye),
FOREIGN KEY(IDListe) REFERENCES Liste(IDListe)
```

```
);

-- Table reviser CREATE TABLE reviser(
mailEmploye VARCHAR(50),
IDListe INT,
PRIMARY KEY(mailEmploye, IDListe),
FOREIGN KEY(mailEmploye) REFERENCES Employe(mailEmploye),
FOREIGN KEY(IDListe) REFERENCES Liste(IDListe)
);

-- Table Changer CREATE TABLE Changer(
mailEmploye VARCHAR(50),
IDTache INT,
PRIMARY KEY(mailEmploye, IDTache),
FOREIGN KEY(mailEmploye) REFERENCES Employe(mailEmploye),
FOREIGN KEY(IDTache) REFERENCES Tache(IDTache)
);

-- Table gerer CREATE TABLE gerer(
mailEmploye VARCHAR(50),
 IDCategorie INT,
PRIMARY KEY(mailEmploye, IDCategorie),
FOREIGN KEY(mailEmploye) REFERENCES Employe(mailEmploye),
FOREIGN KEY(IDCategorie) REFERENCES Categorie(IDCategorie)
);

-- Table completer (pour suivre qui a complété une tâche) CREATE TABLE
completer(
mailEmploye VARCHAR(50),
IDTache INT,
dateCompletion DATETIME NOT NULL,
PRIMARY KEY(mailEmploye, IDTache),
FOREIGN KEY(mailEmploye) REFERENCES Employe(mailEmploye),
FOREIGN KEY(IDTache) REFERENCES Tache(IDTache)
);

-- Trigger pour validation des listes personnelles avant insertion
DELIMITER //
CREATE TRIGGER before_liste_insert
BEFORE INSERT ON Liste
FOR EACH ROW
BEGIN
IF NEW.estPersonnelle = TRUE AND NEW.IDCategorie IS NOT NULL THEN
```

```
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Une liste personnelle ne peut pas être associée à une catégorie';
END IF;
IF NEW.estPersonnelle = FALSE AND NEW.IDCategorie IS NULL THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Une liste non-personnelle doit être associée à une catégorie';
END IF;
END//
DELIMITER ;

-- Trigger pour validation des listes personnelles avant mise à jour
DELIMITER //
CREATE TRIGGER before_liste_update
BEFORE UPDATE ON Liste
FOR EACH ROW
BEGIN
IF NEW.estPersonnelle = TRUE AND NEW.IDCategorie IS NOT NULL THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Une liste personnelle ne peut pas être associée à une catégorie';
END IF;
IF NEW.estPersonnelle = FALSE AND NEW.IDCategorie IS NULL THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Une liste non-personnelle doit être associée à une catégorie';
END IF;
END//
DELIMITER ;

-- Trigger pour mettre à jour la date de dernière modification d'une liste
DELIMITER //
CREATE TRIGGER after_tache_update
AFTER UPDATE ON Tache
FOR EACH ROW
BEGIN UPDATE Liste
SET dateMAJList = NOW()
WHERE IDListe = NEW.IDListe;
END//
DELIMITER ;

-- Trigger pour mettre à jour la date de complétion d'une tâche
```

```
DELIMITER //
CREATE TRIGGER before_tache_completion
BEFORE UPDATE ON Tache
FOR EACH ROW
BEGIN
IF NEW.etatTache = TRUE AND OLD.etatTache = FALSE THEN
SET NEW.dateCompletionTache = NOW();

-- Insérer dans la table compléter
INSERT INTO completer(mailEmploye, IDTache, dateCompletion)
VALUES (NEW.mailEmploye_1, NEW.IDTache, NOW()
);
END IF;
END//
DELIMITER ;

-- Trigger pour vérifier si une liste doit être archivée
DELIMITER //
CREATE TRIGGER after_tache_complete
AFTER UPDATE ON Tache
FOR EACH ROW
BEGIN
DECLARE toutes_completes BOOLEAN;
DECLARE id_liste_concernee INT;
IF NEW.etatTache = TRUE AND OLD.etatTache = FALSE THEN
SET id_liste_concernee = NEW.IDListe;

-- Vérifier si toutes les tâches de la liste sont complétées
SELECT NOT EXISTS (
SELECT 1 FROM Tache
WHERE IDListe = id_liste_concernee
AND etatTache = FALSE
) INTO toutes_completes;

-- Si toutes les tâches sont complétées, archiver la liste
IF toutes_completes = TRUE THEN

UPDATE Liste
SET estArchivee = TRUE,
dateArchivageListe = NOW()
WHERE IDListe = id_liste_concernee;
END IF;
END IF;
```

END//
DELIMITER ;

Donnees pour test :

H. *Données de connexion*

- **Admins**

Mail : test1@organizheur.com

Mdp: Test123

- **Utilisateurs**

Mail : test2@organizheur.com

Mdp: Test123

I. *Lien vers les repositories :*

- Lien vers le frontend

https://github.com/JeanAuryel/organizheur-frontend.git

- Lien vers le backend

https://github.com/JeanAuryel/organizheur-backend.git