

Projet e-Hôtels

Rapport de Projet

CSI 2532- Base de données I

Hiver 2024

École de Génie Électrique et Science Informatique

Université d'Ottawa

Professeur : Kalonji Kalala H

Groupe 44 :

Étudiant 1 : Worku, Girum 300305611

Étudiant 2 : Dosso, Yacine300277181

Introduction

Ce rapport présente la conception et l'implémentation d'une application de réservation d'Hotels nommée e-Hotels. L'objectif principal de l'application est de permettre aux clients de cinq chaînes hôtelières de renom, avec des emplacements étendus en Amérique du Nord, de réserver facilement des chambres dans leurs hôtels respectifs, tout en offrant une vue en temps réel de la disponibilité des chambres. Cette application vise à simplifier le processus de réservation d'hôtels en intégrant toutes les informations pertinentes sur les chaînes hôtelières, les hôtels, les chambres, les clients et les employés dans une base de données centralisée et conviviale.

Dans ce rapport, nous allons d'abord présenter l'implémentation de la base de données en utilisant PostgreSQL, ainsi que les requêtes SQL et les déclencheurs (triggers) nécessaires pour soutenir les opérations de gestion des données.

Implémentation de la Base de Données

Pour l'implémentation de notre Base de Données, nous avons utilisé PostgreSQL avec les requêtes SQL CREATE avec leur Triggers suivante :

```
CREATE TABLE IF NOT EXISTS public.chambre
(
    id integer NOT NULL,
    price integer NOT NULL,
    commodity text COLLATE pg_catalog."default",
    capacity integer,
    view text COLLATE pg_catalog."default",
    extension text COLLATE pg_catalog."default",
    problems text COLLATE pg_catalog."default",
    hotel_id integer NOT NULL,
    room_num integer NOT NULL,
    floor integer NOT NULL,
    hotelchain_id integer NOT NULL,
    room_type text COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT chambre_pkey PRIMARY KEY (id),
    CONSTRAINT uc_chambre_number UNIQUE (room_num, hotel_id),
    CONSTRAINT rooms_hotel_chain_id_fkey FOREIGN KEY (hotelchain_id)
        REFERENCES public.hotel_chain (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT "rooms_hotel_id_fkey " FOREIGN KEY (hotel_id)
        REFERENCES public.hotel (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.chambre
    OWNER to postgres;
-- Index: idx_chambre_price

-- DROP INDEX IF EXISTS public.idx_chambre_price;

CREATE INDEX IF NOT EXISTS idx_chambre_price
    ON public.chambre USING btree
    (price ASC NULLS LAST)
    TABLESPACE pg_default;

-- Trigger: check_price_before_insert_or_update

-- DROP TRIGGER IF EXISTS check_price_before_insert_or_update ON public.chambre;

CREATE OR REPLACE TRIGGER check_price_before_insert_or_update
    BEFORE INSERT OR UPDATE
    ON public.chambre
    FOR EACH ROW
    EXECUTE FUNCTION public.check_price_positive();
```

```

CREATE TABLE IF NOT EXISTS public.client
(
    id integer NOT NULL DEFAULT nextval('client_id_seq'::regclass),
    first_name text COLLATE pg_catalog."default" NOT NULL,
    last_name text COLLATE pg_catalog."default" NOT NULL,
    address text COLLATE pg_catalog."default",
    created_at timestamp without time zone NOT NULL DEFAULT CURRENT_TIMESTAMP,
    date_enreg timestamp without time zone NOT NULL DEFAULT CURRENT_TIMESTAMP,
    nas integer,
    CONSTRAINT client_pkey PRIMARY KEY (id)
)

```

```

TABLESPACE pg_default;

```

```

ALTER TABLE IF EXISTS public.client
    OWNER to postgres;

```

```

CREATE TABLE IF NOT EXISTS public.employees
(
    id integer NOT NULL DEFAULT nextval('employees_id_seq'::regclass),
    first_name text COLLATE pg_catalog."default" NOT NULL,
    last_name text COLLATE pg_catalog."default" NOT NULL,
    role text COLLATE pg_catalog."default" NOT NULL,
    email text COLLATE pg_catalog."default" NOT NULL,
    created_at timestamp without time zone NOT NULL DEFAULT CURRENT_TIMESTAMP,
    hotel_id integer NOT NULL,
    nas integer NOT NULL,
    num_tel bigint,
    CONSTRAINT employees_pkey PRIMARY KEY (id),
    CONSTRAINT employees_hotel_id_fkey FOREIGN KEY (hotel_id)
        REFERENCES public.hotel (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

```

```

TABLESPACE pg_default;

```

```

ALTER TABLE IF EXISTS public.employees
    OWNER to postgres;

```

```

-- Index: fki_m

```

```

-- DROP INDEX IF EXISTS public.fki_m;

```

```

CREATE INDEX IF NOT EXISTS fki_m
    ON public.employees USING btree
    (id ASC NULLS LAST)
    TABLESPACE pg_default;

```

```
CREATE TABLE IF NOT EXISTS public.gestionnaire
(
    id integer NOT NULL DEFAULT nextval('gestionnaire_id_seq'::regclass),
    first_name text COLLATE pg_catalog."default" NOT NULL,
    "last_ name" text COLLATE pg_catalog."default" NOT NULL,
    address text COLLATE pg_catalog."default",
    nas integer,
    CONSTRAINT gestionnaire_pkey PRIMARY KEY (id)
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS public.gestionnaire
    OWNER to postgres;
```

```
CREATE TABLE IF NOT EXISTS public.hotel
(
    id integer NOT NULL DEFAULT 'hotel_id_seq'::regclass,
    address text COLLATE pg_catalog."default" NOT NULL,
    number_rooms integer NOT NULL,
    num_tel character varying(20)[] COLLATE pg_catalog."default" NOT NULL,
    email character varying(100)[] COLLATE pg_catalog."default" NOT NULL,
    hotel_chain_id integer NOT NULL,
    category text COLLATE pg_catalog."default",
    nom_hotel text COLLATE pg_catalog."default",
    gestionnaire_id integer NOT NULL,
    CONSTRAINT "Hotel_pkey" PRIMARY KEY (id),
    CONSTRAINT hotel_gestionnaire_id_fkey FOREIGN KEY (gestionnaire_id)
        REFERENCES public.gestionnaire (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT hotels_hotel_chain_id_fkey FOREIGN KEY (hotel_chain_id)
        REFERENCES public.hotel_chain (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.hotel
    OWNER to postgres;
-- Index: idx_hotel_hotel_chain_id

-- DROP INDEX IF EXISTS public.idx_hotel_hotel_chain_id;

CREATE INDEX IF NOT EXISTS idx_hotel_hotel_chain_id
    ON public.hotel USING btree
    (hotel_chain_id ASC NULLS LAST)
    TABLESPACE pg_default;

-- Trigger: trigger_delete_associated_rooms_and_reservations

-- DROP TRIGGER IF EXISTS trigger_delete_associated_rooms_and_reservations ON public.hotel;

CREATE OR REPLACE TRIGGER trigger_delete_associated_rooms_and_reservations
    BEFORE DELETE
    ON public.hotel
    FOR EACH ROW
    EXECUTE FUNCTION public.delete_associated_rooms_and_reservations();
```

```

CREATE TABLE IF NOT EXISTS public.hotel_chain
(
    id integer NOT NULL DEFAULT nextval('hotelchain_id_seq'::regclass),
    hotelchain_name text COLLATE pg_catalog."default" NOT NULL,
    address text COLLATE pg_catalog."default" NOT NULL,
    number_of_hotels integer NOT NULL,
    email character_varying(100)[] COLLATE pg_catalog."default" NOT NULL,
    num_tel character_varying(20)[] COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT hotel_chain_pkey PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.hotel_chain
    OWNER to postgres;

-- Trigger: delete_chain_cascade

-- DROP TRIGGER IF EXISTS delete_chain_cascade ON public.hotel_chain;

CREATE OR REPLACE TRIGGER delete_chain_cascade
    AFTER DELETE
    ON public.hotel_chain
    FOR EACH ROW
    EXECUTE FUNCTION public.delete_hotels_and_rooms();

```

```

CREATE TABLE IF NOT EXISTS public.location
(
    id integer NOT NULL DEFAULT nextval('location_id_seq'::regclass),
    date_arrive timestamp without time zone NOT NULL,
    date_depart timestamp without time zone NOT NULL,
    paiement integer NOT NULL,
    chambre_num integer NOT NULL,
    created_at timestamp without time zone NOT NULL DEFAULT CURRENT_TIMESTAMP,
    employee_id integer,
    client_id integer,
    CONSTRAINT "Location_pkey" PRIMARY KEY (id),
    CONSTRAINT location_chambre_id_fkey FOREIGN KEY (chambre_num)
        REFERENCES public.chambre (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT location_client_id_fkey FOREIGN KEY (client_id)
        REFERENCES public.client (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT location_employee_id_fkey FOREIGN KEY (employee_id)
        REFERENCES public.employees (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.location
    OWNER to postgres;

```

```

CREATE TABLE IF NOT EXISTS public.reservations
(
    id integer NOT NULL DEFAULT nextval('reservation_id_seq'::regclass),
    arriving_date timestamp without time zone NOT NULL,
    departing_date timestamp without time zone NOT NULL,
    reservation_date timestamp without time zone,
    client_id integer NOT NULL,
    created_at timestamp without time zone NOT NULL DEFAULT CURRENT_TIMESTAMP,
    room_id integer NOT NULL,
    CONSTRAINT "reservations _pkey" PRIMARY KEY (id),
    CONSTRAINT reservations_client_id_fkey FOREIGN KEY (client_id)
        REFERENCES public.client (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT reservations_room_id_fkey FOREIGN KEY (room_id)
        REFERENCES public.chambre (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.reservations
    OWNER to postgres;
-- Index: idx_reservations_client_id

-- DROP INDEX IF EXISTS public.idx_reservations_client_id;

CREATE INDEX IF NOT EXISTS idx_reservations_client_id
    ON public.reservations USING btree
    (client_id ASC NULLS LAST)
    TABLESPACE pg_default;

```

```

--Index sur la table des réservations par client_id
CREATE INDEX idx_reservations_client_id ON public."reservations "(client_id);

--Sur le prix des chambres pour accélérer les recherches par prix :
CREATE INDEX idx_chambre_price ON public.chambre(price);

--Sur l'ID de la chaîne hôtelière dans la table hôtel pour les jointures :
CREATE INDEX idx_hotel_hotel_chain_id ON public."Hotel"(hotel_chain_id);

```

INDEX JUSTIFICATION CHOIX

1. Index sur la table des réservations par client_id: suivi des réservations

- Cet index est utile pour accélérer les requêtes impliquant la recherche de réservations spécifiques pour un client donné. Par exemple, lorsqu'un

utilisateur souhaite voir toutes ses réservations passées, la recherche par **client_id** sera fréquente. L'index permettra ainsi d'optimiser les performances de telles requêtes en réduisant le temps nécessaire pour localiser les enregistrements correspondants.

- Le type de mise à jour attendues au niveau de la base de données sont essentiellement l'ajout de nouvelles réservations, modification ou annulation des réservations existantes.

2. Index sur le prix des chambres pour accélérer les recherches par prix:

- L'utilité de cet index est visible lorsqu'il s'agit des requêtes qui concernent des filtres basés sur le prix des chambres. Par exemple, lorsqu'un utilisateur recherche des chambres dans une certaine gamme de prix, cet index permettra une récupération plus rapide des enregistrements correspondants en évitant une exploration exhaustive de toutes les entrées de la table.
- Le type de mise à jour attendues au niveau de la base de données sont par exemple la modification des prix des chambres.

3. Index sur l'ID de la chaîne hôtelière dans la table hôtel pour les jointures:

- C'est l'index est important pour l'accélération des opérations de jointure qui implique la table des hôtels et d'autres tables où l'ID de la chaîne hôtelière est utilisé comme clé étrangère. L'index accélérera la recherche des enregistrements dans la table des hôtels en fonction de l'ID de la chaîne hôtelière, ce qui améliorera les performances des jointures impliquant cette colonne.
- Le type de mises à jour de données attendues sont par exemple la modification des détails de l'hôtel, tels que les coordonnées ou les informations d'affiliation à une chaîne hôtelière.

Ainsi, ces index ont été conçus dans le but d'accélérer des types spécifiques de requêtes fréquemment exécutées dans la base de données, en réduisant le temps nécessaire pour rechercher et récupérer les données correspondantes. Cela permet une amélioration globale des performances du système, en particulier dans les cas où les tables sont grandes et les requêtes sont assez complexes, permettant ainsi de favoriser l'expérience utilisateur. Cependant, il est important de considérer les mises à jour attendues de la base de données, telles que celles listées ci-dessus. En effet, chaque fois qu'une donnée indexée est modifiée, l'index correspondant doit également être mis à jour. Ceci,

pourrait entraîner un coût supplémentaire au niveau des performances et de l'utilisation de l'espace.

Implémentation de l'application

Pour l'implémentation de l'interface utilisateur de l'application, nous avons utilisé PHP pour établir la connexion avec la base de données :

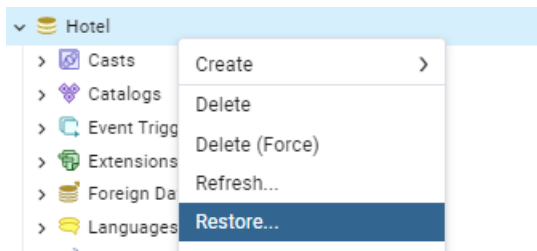
```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    try {
        // Database connection parameters
        $host = 'localhost';
        $port = '5432';
        $dbname = 'Hotel';
        $user = 'postgres';
        $password = 'abc123';

        // Establish database connection
        $dsn = "pgsql:host=$host;port=$port;dbname=$dbname;user=$user;password=$password";
        $conn = new PDO($dsn);
```

Pour la partie front-end, nous avons utilisé le langage HTML pour la structure et le contenu de la page web, ainsi que CSS pour le style et la mise en forme. En combinant HTML et CSS, nous avons pu concevoir une interface utilisateur attrayante et conviviale, offrant une expérience utilisateur optimale lors de la navigation et de l'utilisation de l'application

Guide pour l'installation de l'application

1. Nous avons mis dans le fichier zip un fichier SQL « XXXXXX » qu'il faudrait importer dans votre serveur pgadmin 4, pour ensuite avoir accès à notre base de données.

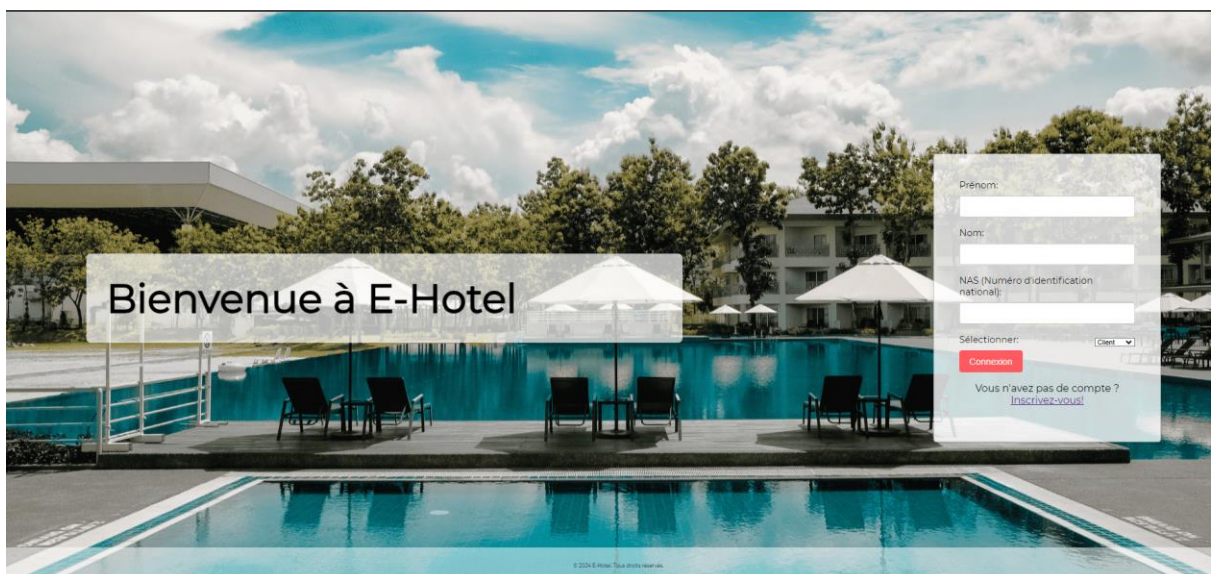


Ensuite dans tout les fichier php il faut changer les informations de votre serveur :

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    try {

        $host = 'localhost';
        $port = '5432';
        $dbname = 'Hotel';
        $user = 'postgres';
        $password = 'abc123';
```

2. Il faut mettre tout les fichier .php ainsi que le dossier « Pictures » (tous dans le zip) dans le repertoire httpd-2.4.58-240131-win64-VS17\Apache24\htdocs pour visualiser l'interface de l'utilisateur.
3. Pour commencer vous devriez ouvrir la page login. Vous pouvez créer un compte ou utiliser ceux-ci :



First Name:

Last Name:

NAS (National Identification Number):

Select: Client ▼

Login

Don't have an account? [Sign up!](#)

First Name:

Last Name:

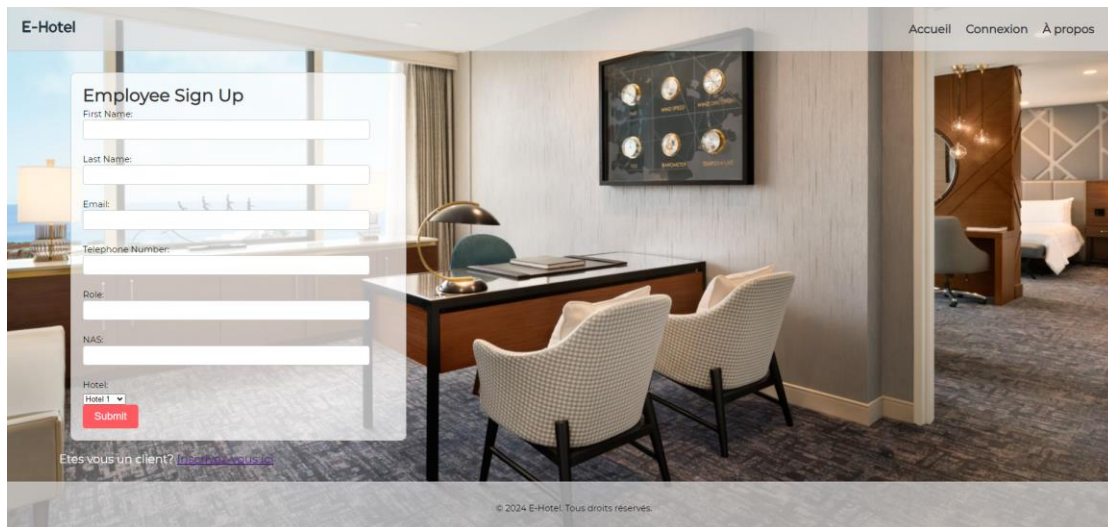
NAS (National Identification Number):

Select: Employee ▼

Login

Don't have an account? [Sign up!](#)





4. Client :

Après avoir créer un compte et login, vous accédez aux pages suivantes :



E-Hotel

[Home](#)[Login](#)[About](#)


Chaine hôtelière:

Nom de l'hôtel:


Capacité:

Prix (max):


Rechercher






Chaine Hôtelière: Chain A
Nom de l'Hôtel: Hotel 1
Numéro de Chambre: 102
Capacité: 2
Vue: Garden View
Prix: 150



Chaine Hôtelière: Chain A
Nom de l'Hôtel: Hotel 1
Numéro de Chambre: 103
Capacité: 2
Vue: Sea View
Prix: 200



Chaine Hôtelière: Chain A
Nom de l'Hôtel: Hotel 1
Numéro de Chambre: 104
Capacité: 1
Vue: City View
Prix: 250




En cliquant sur une des chambres vous pourrez voir ces informations et effectuée une réservation :

E-Hotel

[Home](#)[Login](#)[About](#)

Détails de la Chambre

Chaine hôtelière: Chain A
Nom de l'Hôtel: Hotel 1
Numéro de Chambre: 103
Capacité: 2
Type de Chambre: Suite
Vue: Sea View
Commodité: Wi-Fi, TV, Mini-bar, Balcony
Problèmes: None
Prix: 200



Réserver cette chambre

Date d'arrivée:

Date de départ:

Rechercher

© 2024 E-Hotel. Tous droits réservés

Une fois la réservation effectuée vous pouvez voir la liste de réservation don votre profil :

E-Hotel

AccueilConnexionÀ propos



Informations du Client

Prénom: Gary

Nom: Galaxy

Adresse: 666 Customer Avenue, City P

Réervations du Client

Chaîne Hôtelière: Chain A

Nom de l'Hôtel: Hotel 1

Type de Chambre: Suite

Date d'arrivée: 2024-04-15 14:38:00

Date de départ: 2024-04-19 14:38:00


© 2024 E-Hotel. Tous droits réservés.

Employée :

Vous pouvez voir la liste de réservations des clients et les changer en location :

E-Hotel

AccueilConnexionÀ propos



Informations de l'Employé

Prénom: Test

Nom: Test

Email: Test

Numéro de Téléphone: 465798132

Rôle: Gestion

Nom de l'Hôtel: Hotel 15

Réervations des Clients

Prénom: Bob

Nom: Builder

ID de Réservation: 1

Date d'arrivée: 2024-04-03 02:28:00

Date de départ: 2024-04-04 02:28:00

Date de réservation: 2024-04-03 02:28:49.525835

Enregistrer Location

Prénom: Bob

Nom: Builder

ID de Réservation: 2

Date d'arrivée: 2024-04-05 03:09:00

E-Hotel

AccueilConnexionÀ propos

Enregistrer Location

Paiement (numero de carte) :

Enregistrer Location

Veuillez remplir le champ de paiement.