



Gestion des utilisateurs

Ajout de la table **users**

```
CREATE TABLE "users" (  
    "id"      INTEGER NOT NULL,  
    "prenom"  TEXT NOT NULL,  
    "password" TEXT NOT NULL,  
    PRIMARY KEY("id" AUTOINCREMENT)  
);
```

factbook_users.sqlite

Création d'utilisateurs



Création d'utilisateurs: modèle

```
# models/users.py

from ..app import app, db

class Users(db.Model):
    __tablename__ = "users"

    id = db.Column(db.Integer, unique=True, nullable=False, primary_key=True, autoincrement=True)
    prenom = db.Column(db.Text, nullable=False)
    password = db.Column(db.String(100), nullable=False)
```

Création d'utilisateurs: méthode statique

Gestion des utilisateurs

```
# models/users.py
from werkzeug.security import generate_password_hash
class Users(db.Model):
    ...
    @staticmethod
    def ajout(prenom, password):
        erreurs = []
        if not prenom:
            erreurs.append("Le prénom est vide")
        if not password or len(password) < 6:
            erreurs.append("Le mot de passe est vide ou trop court")
        unique = Users.query.filter(
            db.or_(Users.prenom == prenom)).count()
        if unique > 0:
            erreurs.append("Le prénom existe déjà")
```

Création d'utilisateurs: méthode statique

```
if len(erreurs) > 0:
    return False, erreurs
utilisateur = Users(
    prenom=prenom,
    password=generate_password_hash(password))

try:
    db.session.add(utilisateur)
    db.session.commit()
    return True, utilisateur
except Exception as erreur:
    return False, [str(erreur)]
```

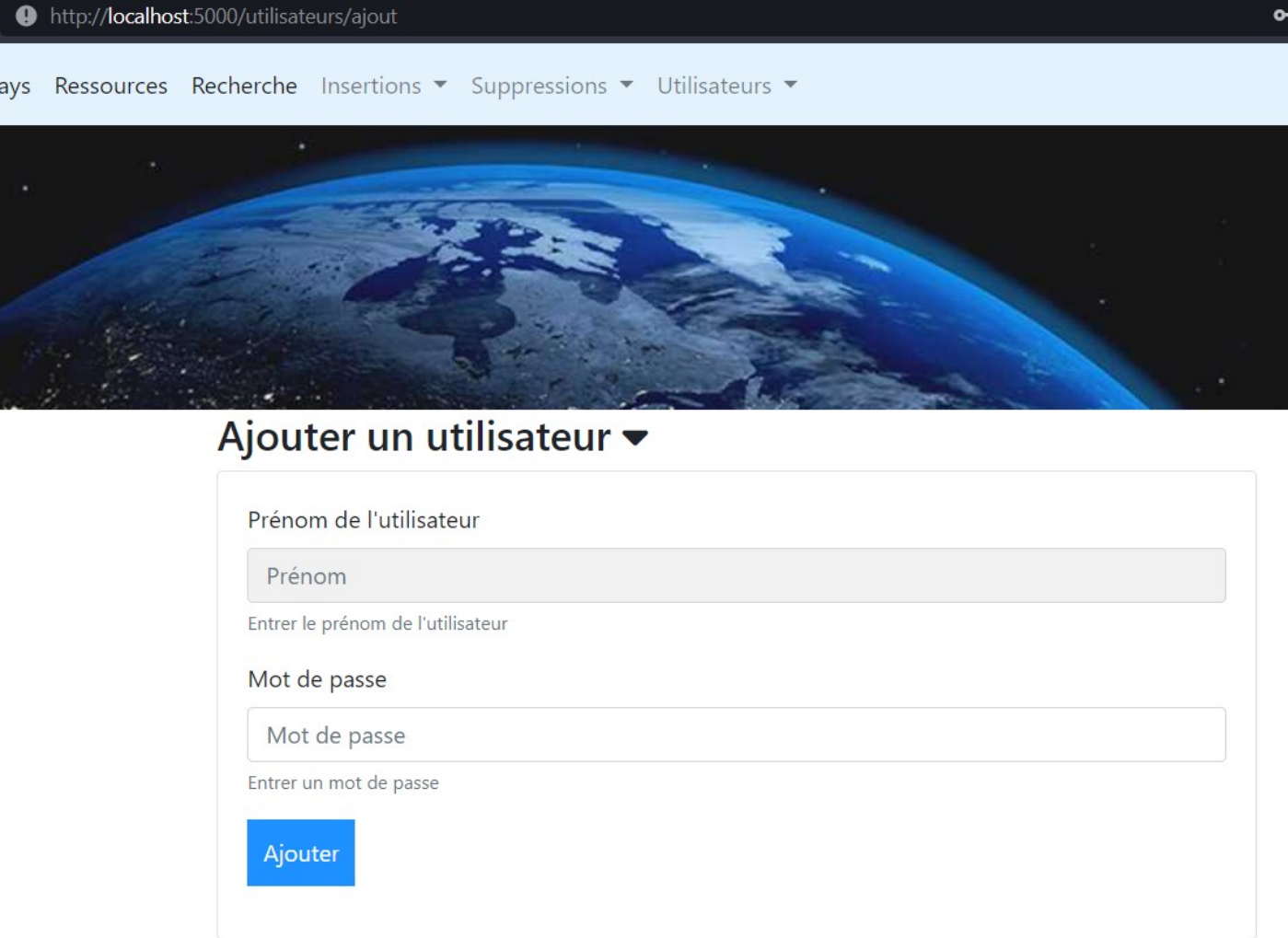
Création d'un utilisateur: variable d'environnement

```
# .env  
  
SECRET_KEY=duseInondevinableetunique
```

```
# config.py  
  
...  
class Config():  
    ...  
    SECRET_KEY = os.environ.get("SECRET_KEY")
```


Création d'un utilisateur

Gestion des utilisateurs



http://localhost:5000/utilisateurs/ajout

ays Ressources Recherche Insertions ▼ Suppressions ▼ Utilisateurs ▼



Ajouter un utilisateur ▼



Prénom de l'utilisateur

Entrer le prénom de l'utilisateur

Mot de passe

Entrer un mot de passe

Ajouter

Maxime Challon --  EnC --  Atos

Création d'un utilisateur: FlaskForm

```
# models/formulaires.py

class AjoutUtilisateur(FlaskForm):
    prenom = StringField("prenom", validators=[])
    password = PasswordField("password", validators=[])
```

Création d'un utilisateur: route

Gestion des utilisateurs

```
#routes/users.py

from flask import url_for, render_template, redirect, request, flash
from ..models.users import Users
from ..models.formulaires import AjoutUtilisateur
from ..utils.transformations import clean_arg
from ..app import app, db

@app.route("/utilisateurs/ajout", methods=["GET", "POST"])
def ajout_utilisateur():
    form = AjoutUtilisateur()
    if form.validate_on_submit():
        statut, donnees = Users.ajout(
            prenom=clean_arg(request.form.get("prenom", None)),
            password=clean_arg(request.form.get("password", None)) )
        if statut is True:
            flash("Ajout effectué", "success")
            return redirect(url_for("accueil"))
        else:
            flash(", ".join(donnees), "error")
            return render_template("pages/ajout_utilisateur.html", form=form)
    else:
        return render_template("pages/ajout_utilisateur.html", form=form)
```

Création d'un utilisateur: template du formulaire

```
<!-- templates/partials/formulaires/ajout_utilisateur.html -->

<form action="{{url_for('ajout_utilisateur')}}" method="post" name="ajout_utilisateur">
  {{ form.hidden_tag() }}
  <div class="form-group">
    <label for="prenom">Prénom de l'utilisateur</label>
    {{ form.prenom(class_="form-control", placeholder_="Prénom") }}
    <small id="prenom" class="form-text text-muted">Entrer le prénom de l'utilisateur</small>
  </div>
  <div class="form-group">
    <label for="password">Mot de passe</label>
    {{ form.password(class_="form-control", placeholder_="Mot de passe") }}
    <small id="password" class="form-text text-muted">Entrer un mot de passe</small>
  </div>
  <p><input type="submit" value="Ajouter"></p>
</form>
```

Création d'un utilisateur: template de route

```
<!-- templates/pages/ajout_utilisateur.html -->

<div class="col-sm-8">

  <h3 data-toggle="collapse" data-target="#collapseExample" aria-expanded="false"
    aria-controls="collapseExample">
    Ajouter un utilisateur <i class="fa-solid fa-caret-down"></i>
  </h3>
  <div class="collapse show" id="collapseExample">
    <div class="card card-body">
      {% include "partials/formulaires/ajout_utilisateur.html" %}
    </div>
  </div>
</div>
```

Connexion

```
pip install flask-login
```

Connexion: instancier LoginManger

```
# app/app.py

from flask_login import LoginManager

...
login = LoginManager(app)

from .routes import ...
```

Connexion: héritage de la classe Users

```
# models/users.py

from flask_login import UserMixin

class Users(UserMixin, db.Model):
    ...
```

Connexion: méthodes de classe

Users

```
# models/users.py
from ..app import app, db, login


class Users(UserMixin, db.Model):
    ...
    def get_id(self):
        return self.id

    @login.user_loader
    def get_user_by_id(id):
        return Users.query.get(int(id))
```


Gestion des utilisateurs Connexion: TD

! http://localhost:5000/utilisateurs/connexion

ays Ressources Recherche Insertions ▾ Suppressions ▾ Utilisateurs ▾



Se connecter ▾

Prénom de l'utilisateur

Entrer le prénom

Mot de passe

Entrer un mot de passe

Connexion: TD: méthode

- création de la méthode statique `Users.identification`
- création du FlaskForm `Connexion`
- création de la route `/connexion`
- création des templates nécessaires

Connexion: TD

```
# models/users.py

from werkzeug.security import check_password_hash

class Users(UserMixin, db.Model):
    ...

    @staticmethod
    def identification(prenom, password):
        utilisateur = Users.query.filter(Users.prenom == prenom).first()
        if utilisateur and check_password_hash(utilisateur.password, password):
            return utilisateur
        return None
```

Connexion: TD

```
# models/formulaires.py

class Connexion(FlaskForm):
    prenom = StringField("prenom", validators=[])
    password = PasswordField("password", validators=[])
```

Gestion des utilisateurs Connexion: TD

```
# routes/users.py
from flask_login import login_user, current_user

@app.route("/utilisateurs/connexion", methods=["GET","POST"])
def connexion():
    form = Connexion()
    if current_user.is_authenticated is True:
        flash("Vous êtes déjà connecté", "info")
        return redirect(url_for("accueil"))
    if form.validate_on_submit():
        utilisateur = Users.identification(
            prenom=clean_arg(request.form.get("prenom", None)),
            password=clean_arg(request.form.get("password", None)) )
        if utilisateur:
            flash("Connexion effectuée", "success")
            login_user(utilisateur)
            return redirect(url_for("accueil"))
        else:
            flash("Les identifiants n'ont pas été reconnus", "error")
            return render_template("pages/connexion.html", form=form)
    else:
        return render_template("pages/connexion.html", form=form)

login.login_view = 'connexion'
```

Déconnexion

```
# routes/users.py
from flask_login import current_user, logout_user

@app.route("/utilisateurs/deconnexion", methods=["POST", "GET"])
def deconnexion():
    if current_user.is_authenticated is True:
        logout_user()
    flash("Vous êtes déconnecté", "info")
    return redirect(url_for("accueil"))
```

Déconnexion

```
<!-- partials/conteneur.html -->

{% if current_user.is_authenticated %}
<a class="dropdown-item" href="{{ url_for('deconnexion') }}">Se déconnecter</a>
{% else %}
<a class="dropdown-item" href="{{ url_for('connexion') }}">Se connecter</a>
{% endif %}
```

Restrictions d'accès

```
# routes/generales.py

from flask_login import login_required

@app.route("/recherche", methods=['GET', 'POST'])
@app.route("/recherche/<int:page>", methods=['GET', 'POST'])
@login_required
def recherche(page=1):
    ...
```