UNIVERSITÉ LIBRE DE BRUXELLES

ULB

Faculté de Lettres, Traduction et Communication

Conception et gestion de banques de données : Rapport de projet

Budget Squirrel : Application de gestion de budget

Milena ALBU
Marie HUYSMAN

Rapport de projet écrit dans le cadre du cours STIC-B505 : Conception et gestion de banques de données

Année académique 2020–2021

Table des matières

1 Introduction

Si besoin de citer : (**SOURCE**) Notes de bas de page [1]

Dans le cadre de ce projet, nous avons choisi de développer une application appartenant au domaine de gestion financière et permettant la gestion de budget à une échelle personnelle. Le rapport ci-dessous en documente le processus de design et conception, ainsi que les développements successives, et les améliorations apportées à l'idée de base précédemment décrite. Après une brève présentation du domaine d'application, nous présenterons le schéma conceptuel, ainsi que sa traduction en schéma relationnel. Ensuite, nous décrirons les différents éléments des scripts SQL que nous avons mis en place (clés uniques et étrangères, contraintes au niveau de la base des données, checks et triggers, ainsi que privilèges d'accès) . Enfin, nous exposerons brièvement l'application Web que nous avons développée. Vous trouverez, en annexe à ce rapport, deux scripts sql : `initdb.sql`, qui contient tous les scripts permettant la création de la base de donnée en elle-même, et `data.sql`, qui peuple la base de données de quelques informations, permettant ainsi de parcourir les différentes pages de l'application.

1.1 Domaine d'application

Une application de gestion de budget est une application qui permet, au minimum, l'enregistrement ainsi que la suppression des transactions dans une base des données, la gestion des utilisateurs et de leur budget, et la possibilité d'agréger une vue d'ensemble sur les transactions financières enregistrés. Afin de mettre en oeuvre une telle application les développeurs doivent connaître les besoins minimes d'utilisateurs potentiels, ainsi que les différentes techniques de programmation à employer pour répondre de manière pertinente, voir optimale, à ce besoins. Dans le cas de budgetsquirrel, l'application se focalise sur les utilisateurs qui n'ont pas nécessairement une expertise spécifique avec le planning financier, mais qui souhaitent néanmoins avoir une vue d'ensemble, ainsi qu'un suivi quotidien, de leurs dépenses. L'application est à usage restraint et, dans sa présente version (en usage libre, lié aux profils de ses utilisateurs, mais sans être protégé par un système de mot-de-passe) nous recommandons qu'elle soit employé au sein d'une réseau domestique, et sur un server local. Comme convenu, dans sa dernière version l'application sait gérer plusieurs cas d'utilisation : connexion et déconnexion, création et modification de profil, enregistrement et suppression des transactions, vue historique propriétaire à l'utilisateur, et accessible en mode consultation ainsi qu'en mode édition (permet la suppression des transactions listés), et vue statistique avec visualisation dynamique des données liées aux transac-

1. informations suppl.

tions par mois (bar chart), et par catégorie (pie charts). L'interface de budgetsquirrel compte 7 écrans, divisés sur 4 catégories :

* (1) gestion de profil (connexion, inscription, profil, et logout) ;
* (2) enregistrement (écran d'enregistrement) ;
* (3) historique des transactions (écran historique), et
* (4) statistiques (écran statistiques).

Dans la réalisation d'écrans des éléments de PHP (76.2 %), CSS (6.4 %), HTML (1.9 %), et JAVASCRIPT (0.1%) ont été utilisés. Pour rédiger le rapport nous avons employé le langage LaTeX (15.4 % de notre projet).

2 Schéma conceptuel

Le schéma conceptuel de la base des données budgetsquirrel reste conforme au schéma validé lors de la première étape du projet. Nous comptons 5 tables principales, ainsi que 3 tables issues de la matérialisation d'heritage sur la table transaction financière. Concernant les attributs (mieux détaillés dans la section 3), chaque table en contient au minimum un identifiant (clé primaire), ainsi que des contraintes (clé étrangère, ou bien contraintes d'unicité selon le besoin). La base des données en contient aussi quatre vues (utilisées dans la construction d'historique et des statistiques), soit historique_v (vue qui liste, chronologiquement, l'information lié aux transactions effectuées par un utilisateur), stat_depenses_revenus_mois (vue qui liste, chronologiquement, les dépenses et les revenus enregistrés par l'utilisateur, en faisant également le bilan dépense, respectivement le bilan revenus) stat_cat (vue qui liste la répartition totale des dépenses et des revenus par catégorie), et stat_types (vue qui liste la répartition du budget global par type de transaction employé dans le payement, ainsi que par nombre d'emplois de chaque type de transaction, pour un bilan global des dépenses et revenus). Finalement, l'emploi des vues est aussi particulièrement utile, dans la création des graphes.

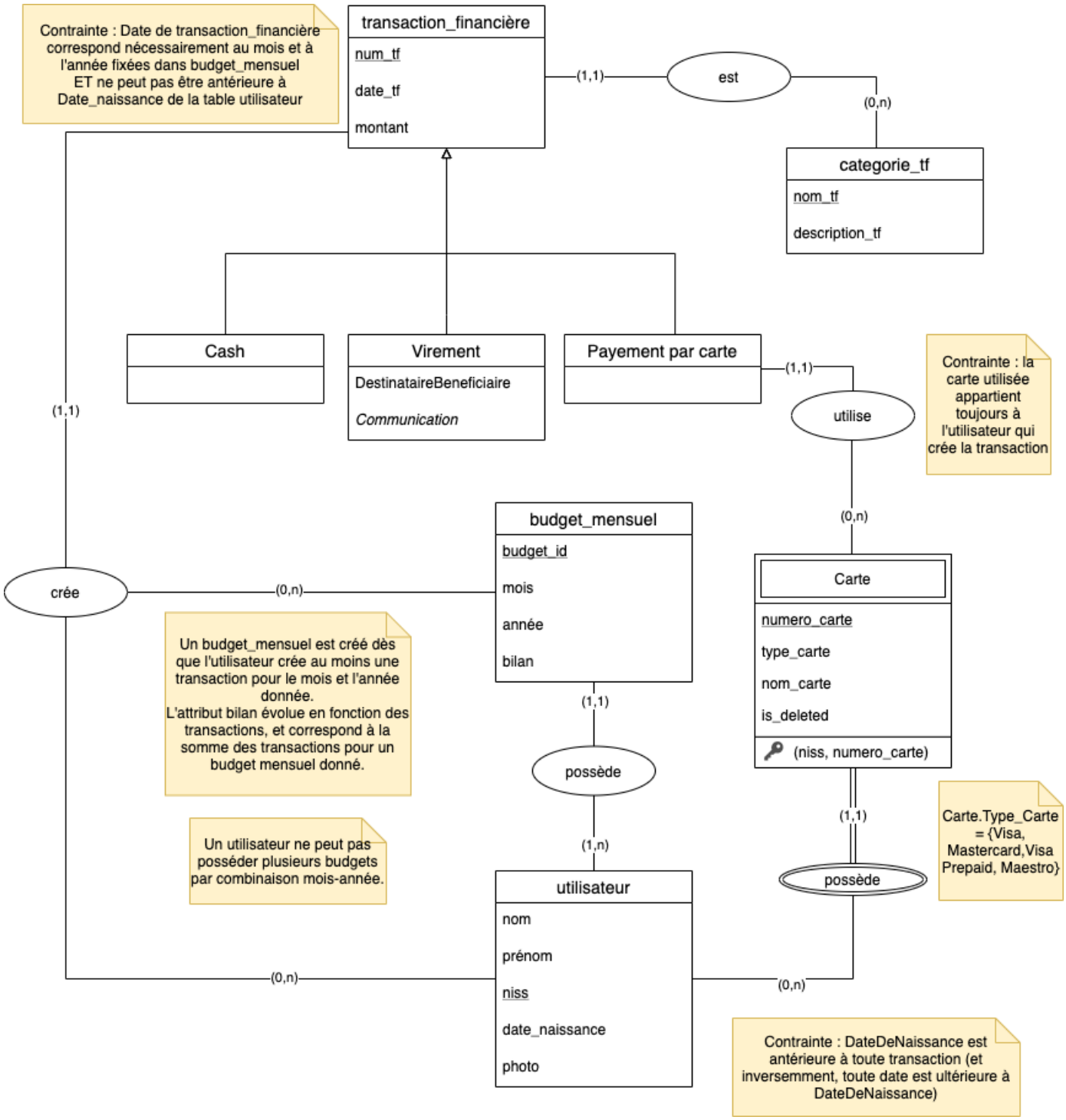

FIGURE 1 – Schéma conceptuel entité-association de la base de données exploitée par Budget Squirrel

3 Schéma logique relationnel

Schéma logique relationnel traduisant le schéma conceptuel, avec une explication du choix des clés de traductions (héritage, associations, contraintes...)

! vérifier si tout correspond avec la DB

! ajouter contraintes

! explication matérialisation

```
utilisateur(nom, prenom, niss, date_naissance, photo, nombre_transactions)

carte(nom_carte, numero_carte, type_carte, niss_util, is_deleted)
carte.niss_util référence utilisateur.niss
carte.type_Carte = {Visa, Mastercard,Visa Prepaid, Maestro}

budget_mensuel(budget_ID, mois, annee, statut, bilan, reste, niss_util)
budget_mensuel.niss_util référence utilisateur.niss

transaction_financiere(num_tf, date_tf, montant, budget_ID, categorie)
transaction_financiere.budget_ID référence budget_mensuel.budget_ID
transaction_financiere.categorie référence categorie.nom_cat

tf_cash(num_tf)
tfcash.num_tf référence transaction_financiere.num_tf

tf_virement(num_tf, communication, destbenef)
tfvirement.num_tf référence transaction_financiere.num_tf

tf_carte(num_tf, nom_carte)
tfcarte.num_tf référence transaction_financiere.num_tf
nom_carte référence carte.nom_carte

categorie_tf(nom_cat, description)
```

4 Code SQL

4.1 Description du code de création de la base de données

Description du code sql de création de la db!! La création de la db doit : créer des tables , avoir des contraintes de clé primaire, unique et externe, avoir des contraintes sous forme de prédicat de table et de colonne, avoir des procédures stockées et des déclencheurs

4.2 Éléments de SQL avancé

4.2.1 Contraintes garanties par des check

4.2.2 Requêtes de consultation

4.2.3 Requêtes de mise à jour

4.2.4 Vues utilisées

Ce qu'on utilise pour l'historique et les différentes statistiques

4.2.5 Déclencheurs

Triggers à utiliser, sans doute, pour le calcul des bilans, et autres

4.2.6 Droits d'accès

Voir si les droits d'accès peuvent être utilisés comme dans le rapport publié par Servais, pour l'instant ça reste un accès root défini par une session > problématique?

5 Description de l'application Web

L'application web que nous avons développé a pour objectif de permettre aux utilisateurs de s'inscrire, d'enregistrer leurs transactions, de consulter leur historique par mois, ainsi que des statistiques sur la répartition de leurs dépenses, de leur revenus, et un bilan total de leur budget. L'accès à la base de donnée est restreint : ce n'est pas un accès root, mais bien un accès spécifiquement défini pour les utilisateurs de l'application. Nous en avions fait la description détaillée à la section précédente.

À l'ouverture de l'application (`localhost/budgetsquirrel/index.html` ou simplement `localhost/budgetsquirrel/`), l'utilisateur a deux possibilités : se connecter, ou s'inscrire.

FIGURE 2 – Landing page pour Budget Squirrel

S'il choisit de s'inscrire, il est redirigé vers un formulaire d'inscription où il doit rentrer son nom, son prénom, son NISS, et choisir une photo de profil.

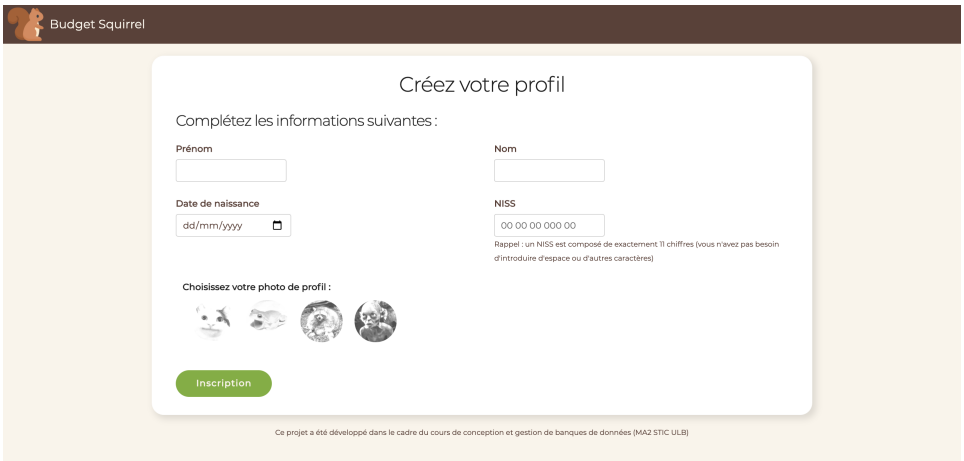

FIGURE 3 – Écran inscription Budget Squirrel

Si un utilisateur est déjà inscrit sous le NISS entré, il en est notifié (et a la possibilité de se rediriger vers la page de connexion). Si le NISS n'est pas encore utilisé par un utilisateur, l'enregistrement se fait, l'utilisateur est notifié du succès de ce dernier, et peut se rediriger vers la page de connexion. La vérification de la présence d'un utilisateur utilisant déjà le NISS se fait à deux niveaux : au niveau de la base de donnée en elle-même, comme nous l'avons vu au dessus, grâce à la contrainte de clé primaire, et également en php, en faisant une sélection de la table utilisateur (si la sélection par rapport au niss entrée renvoie un résultat différent de zéro, l'utilisateur est notifié de l'impossibilité de créer un compte avec ce NISS).

La page de connexion, par souci de simplification, est une simple sélection de profil : l'utilisateur a accès à la liste des utilisateurs, sélectionne son profil (nom et prénom) dans la liste , et se connecte. Ensuite, le NISS de l'utilisateur est récupéré par une méthode POST, enregistré par la méthode SESSION et appelé grâce à `session_start()` ; à chaque page de l'application où il est enregistré comme connecté).

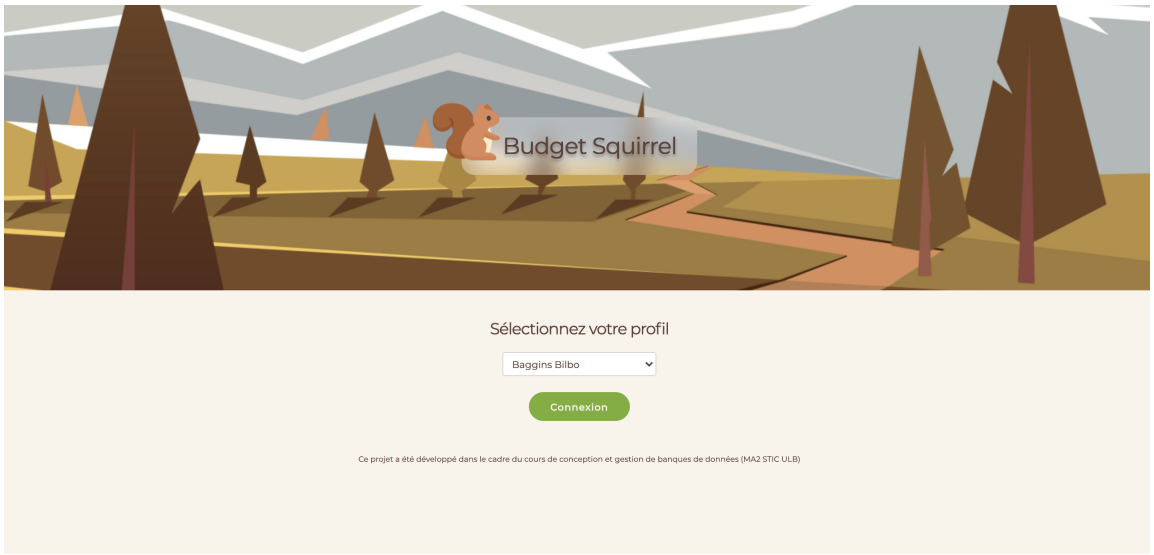

FIGURE 4 – Écran connexion Budget Squirrel

Une fois qu'il a sélectionné son profil et choisi de se connecter, l'utilisateur est redirigé vers une page d'accueil. La page d'accueil présente brièvement les différentes pages de l'application. Depuis la barre de navigation, présente sur toutes les pages (sauf celles ou l'utilisateur n'est pas considéré comme connecté), l'utilisateur a accès : à la page d'accueil, à l'historique, à la page d'enregistrement, à la page de statistiques, et à sa page personnelle de profil. Il peut ainsi naviguer librement entre les différentes pages.

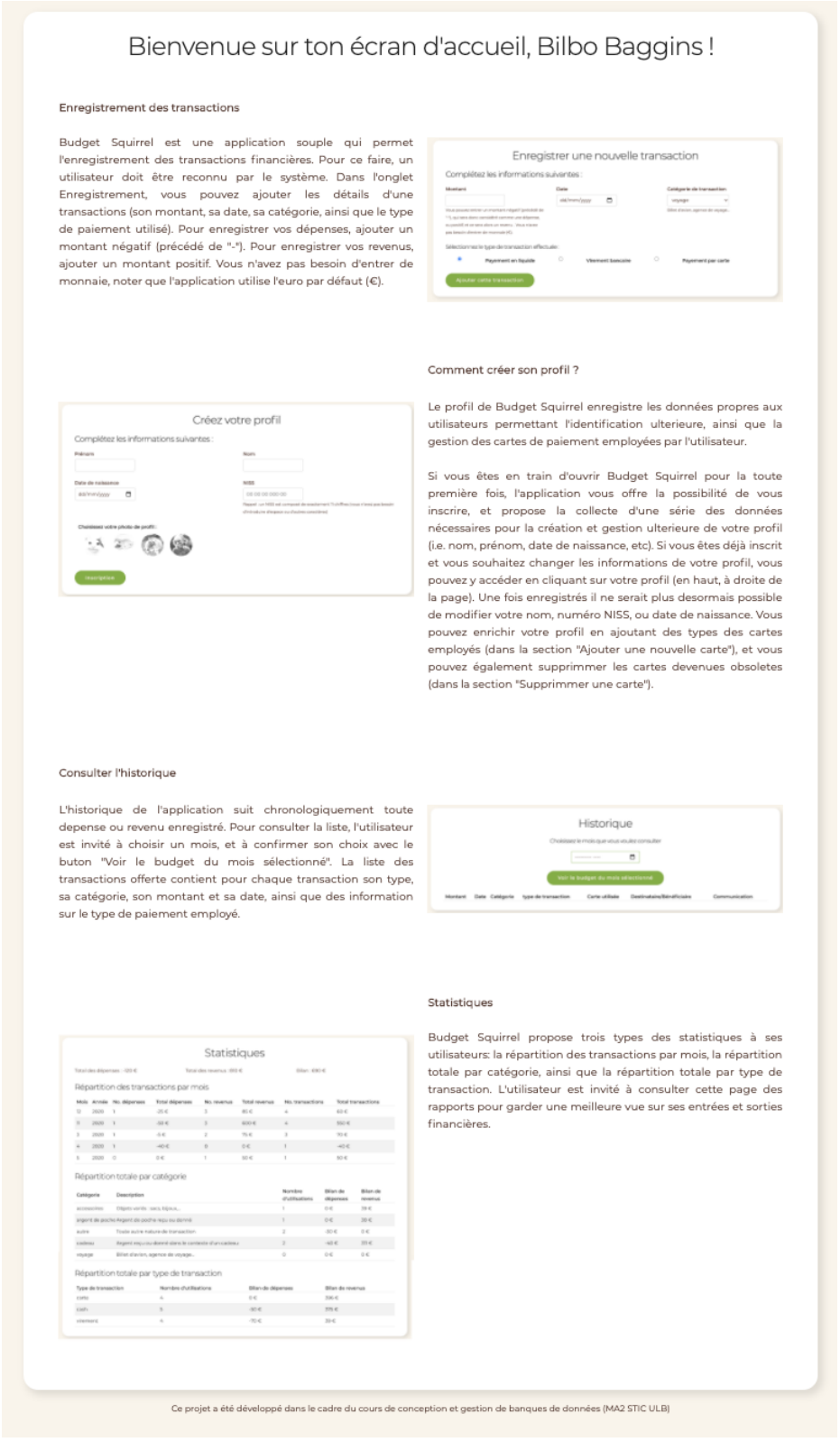

FIGURE 5 – Page d'accueil Budget Squirrel

La page personnelle de profil permet à l'utilisateur de visualiser ses informations : nom, prénom, NISS, date de naissance. Il peut également y ajouter et y supprimer ses cartes [2] . Les listes de ses cartes disponibles et de ses anciennes cartes sont également visibles depuis la page de profil. Il peut également y actualiser sa photo.

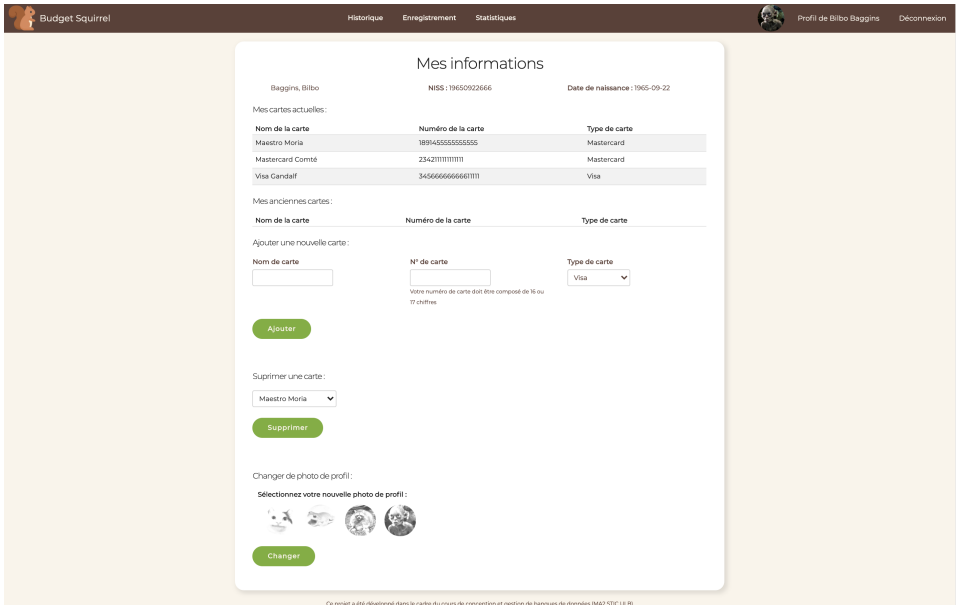

FIGURE 6 – Page personnelle Budget Squirrel

La page d'enregistrement permet d'enregistrer des transactions. L'utilisateur est obligé d'entrer un montant, de sélectionner une date, une catégorie de transaction, et un type de transaction (à nouveau, la vérification se fait une fois en php, et une fois aussi du côté de la base de données). En fonction du type de transaction effectuée, il doit également ajouter des informations supplémentaires. Le payement en liquide ne demande aucune information supplémentaire, le virement bancaire demande d'introduire un destinataire ou un bénéficiaire, et éventuellement une communication, et un payement par carte nécessite de sélectionner la carte utilisée. Cet aspect est géré uniquement en php, et demande donc de faire deux opérations consécutives si l'on veut rentrer le type de transaction sans passer par l'application Web.

2. Cette suppression est une suppression logique du côté de la base de données : les cartes possèdent une colonne is_deleted , par défaut la valeur de la colonne est à 0, et si l'utilisateur « supprime »sa carte, la valeur passe à 1, et la carte est considérée comme virtuellement supprimée

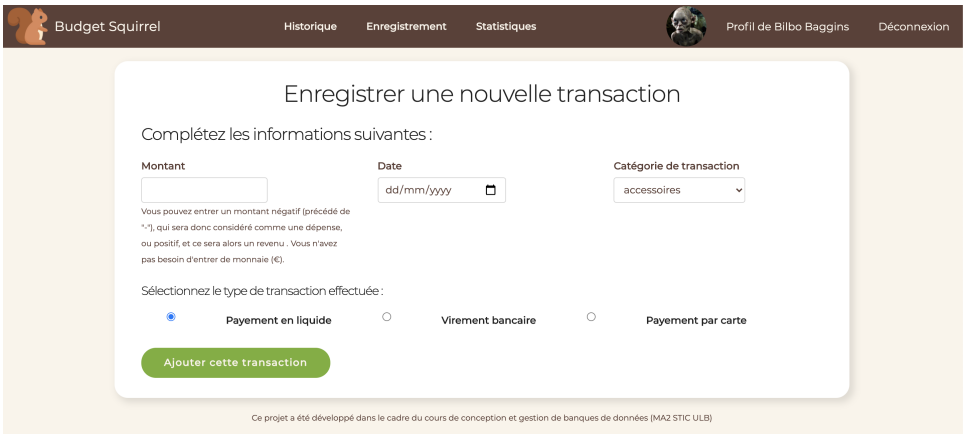

FIGURE 7 – Enregistrement des transactions en Budget Squirrel

La page historique permet à l'utilisateur de visualiser les transactions effectuées : pour cela, il doit d'abord sélectionner le mois et l'année dont il veut consulter le budget. Il a alors accès à un tableau affichant le montant, la date, la catégorie, et le type de transaction effectuée, et éventuellement la carte utilisée, le destinataire/bénéficiaire, et la communication. En dessous de la table, l'utilisateur peut également voir le bilan total du mois. Enfin, il peut également choisir de supprimer une des transactions affichées.

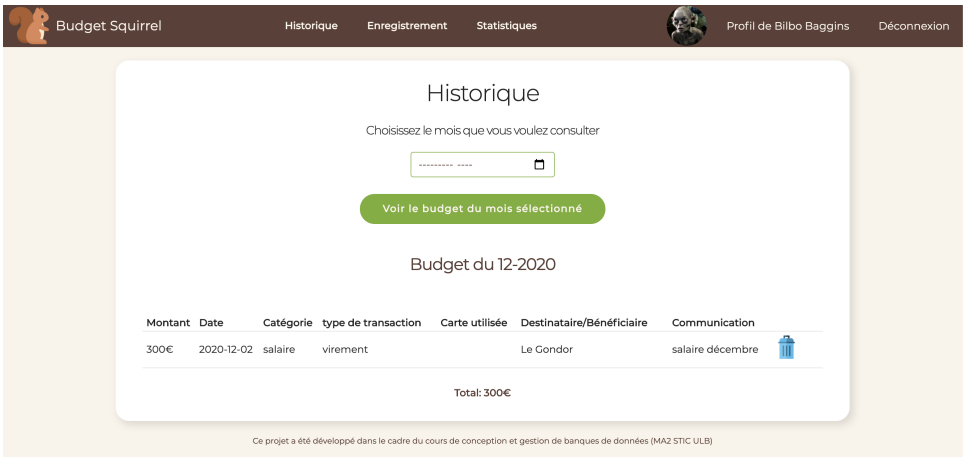

FIGURE 8 – Historique des transactions en Budget Squirrel

L'écran de statistiques offre différentes informations sur les données concernant l'utilisateur. Les informations offertes sont les suivantes : un total des dépenses, un total des revenus, et un bilan, mais aussi un tableau montrant la répartition des transactions (dépenses et revenus) par mois, ainsi que le bilan par mois, pour une vue plus condensée de l'historique, un tableau montrant la répartition totale par catégorie (couplé à une description de chaque catégorie), et enfin la répartition des transactions entre les différents types de payements (le nombre d'utilisations, le bilan des dépenses

et le bilan des revenus pour chaque type de transaction).

À tout moment, l'utilisateur peut également se déconnecter, en cliquant, en haut à droite de la barre de navigation, sur "déconnexion". Les informations de sessions (donc, le NISS sous lequel l'utilisateur est connecté) sont effacées, et l'utilisateur peut choisir de retourner à l'écran d'accueil.

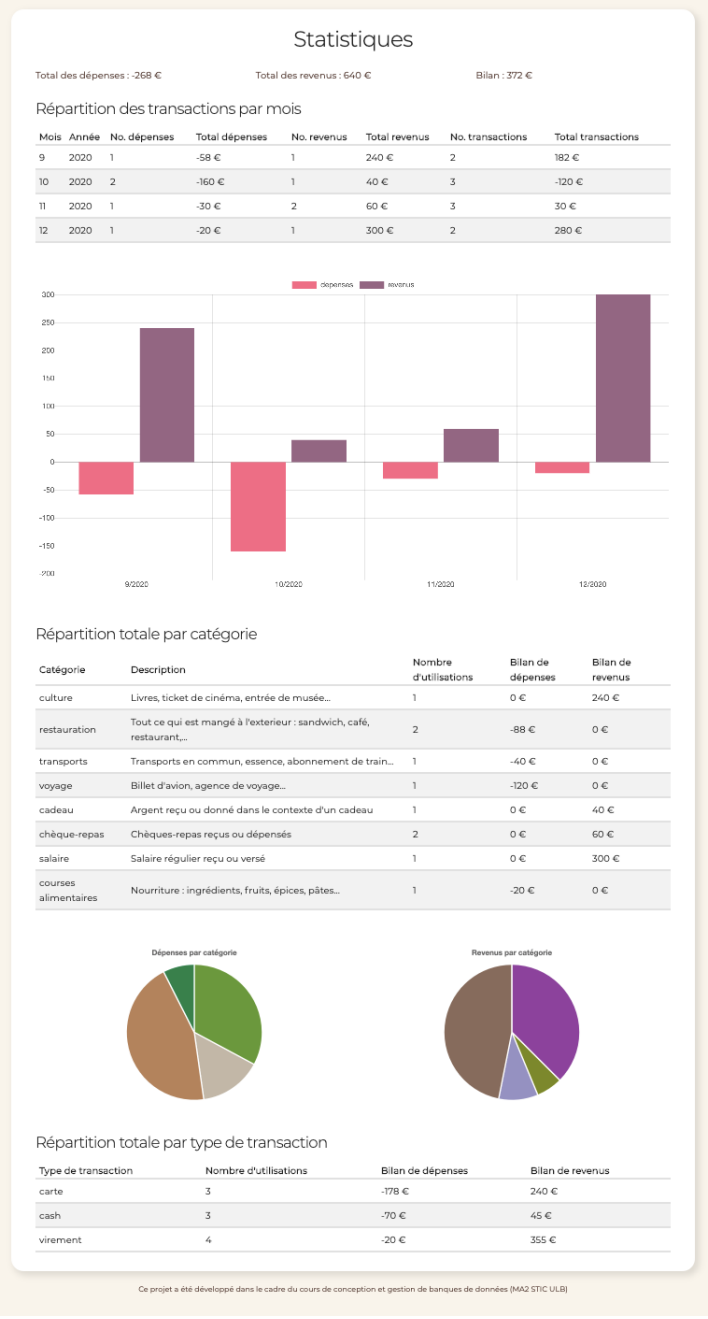

FIGURE 9 – Statistiques Budget Squirrel

6 Développement du projet

Budget Squirrel est une application consistante d'une base des données et une interface client. Le design, développement et administration d'une telle application nécessitent donc une chorégraphie particulière, reposant sur des livrables qui doivent être prêts à l'emploi dans une certaine séquence, et qui se trouvent la plupart de temps dans une symbiose étroite. De plus, certaines exigences du projet, établies ou agréées dès le départ, et révisés pendant les séances d'entretien projet, ainsi que certaines contraintes liées à d'autres engagements impératives pour les développeurs se sont vite imposées comme des conditions demandaient une coordination ainsi qu'une communication ouverte et souple au sein du groupe. Le groupe s'est vite mis d'accord sur une division des tâches qui prend en compte les contraintes relatives aux disponibilités de chacune d'entre nous, ainsi qu'aux préférences des techniques et outils des développement, toute en gardant en vue l'impératif d'avoir touché au moins une fois à chaque écran et surtout d'avoir participé en tandem le plus que possible dans les parties couvrant la matière vue au cours.

6.1 Travail conceptuel et outils employés pour démarrer le projet

En suivant le calendrier communiqué au cours, le groupe à fait sa proposition de projet le 3 octobre, et a reçu une confirmation pour le deuxième sujet dans un bref délai.

Domaine application	Sujet	Utilisateurs cible	Description
Domotique	Gestion de stock de produits dans un frigo	Tout possesseur de frigo intelligent (types d'utilisateurs envisagés: administrateur et utilisateur simple.	**Entités**: frigo, produit, utilisateur. **Contraintes**: - un utilisateur a le droit de consulter les tables, - un administrateur a le droit de modifier les tables. Une **statistique** possible montrera le **taux de remplissage à une date precise, par type de produit** (par rapport à une quantite envisagé).
Planning financier	Système de gestion des dépenses	Personnes privées, banques (types d'utilisateurs envisagés: administrateur, banque, utilisateur simple).	**Entités**: utilisateurs (administrateurs, utilisateurs reguliers, banque), et dépenses (réparties par type de dépense), Une **statistique** possible montrera les dépenses **par type et dans une période de temps**.

FIGURE 10 – Sujets proposés en vue de développement

Une fois le choix de projet validé, nous avons commencé le travail de conceptualisation, avec un double-but : la conception d'un modèle entité-association, ainsi qu'une première ébauche d'interface client. Pour nous aider dans le démarche EA, nous avons employé l'application web draw.io, et en ce qui concerne l'interface client, l'outil Figma à été employé dans la construction des wireframes. Le résultat de ce travail s'est matérialisé dans un rapport préliminaire, présenté pour évaluation et validation le 31 octobre. C'est en suivant ce rapport préliminaire que nous avons par la suite développé les éléments décrits dans les sections 2 et 3 du présent rapport.

6.2 Design et techniques de travail

La partie design à été facilité par l'emploi d'une structure html ouverte, que nous avons taillé aux besoins de notre application. Cette étape à rajouté une première couche de complexité en ce qui concerne l'imbrication des langages utilisés, en ajoutant PHP et HTML aux requêtes SQL. Le PHP étant un langage très flexible, il nous a été relativement facile de créer de nombreuses contraintes, et de combiner différentes requêtes SQL pour que l'application réponde exactement comme il faut à tout input de l'utilisateur. Par contre, un des défis que nous avons rencontré durant le développement de ce projet a été l'écriture de données en utilisant du SQL pur, sans l'aide de l'application. Certains éléments, facilement mis en place en PHP, étaient difficiles à traduire en SQL. Une deuxième couche de complexité à été rajouté par l'introduction des éléments javascript de visualisation dynamique des données (toggle screen, et graphes dynamiques) et des éléments en SQL d'aggregation des données (vues statistiques). De façon général, une des parties la plus difficile à été d'orchestrer, d'une façon robuste et cohérente, la partie d'insertions, mises à jour, et suppressions, afin d'éviter la perte des données, ou bien le double encodage. Faire le choix entre l'ajout des contraintes au niveau de client, ou bien au niveau de la base des données, ainsi que naviguer les contraintes implémentés, n'était pas toujours facile. Pour parer ces difficultés nous avons employé la technique de programmation en binôme (asynchrone, en utilisant GitHub pour partager les mises à jours au niveau du code, et Teams pour faciliter la communication). La taille restreinte du groupe à particulièrement facilité ce méthode de travail, et nous à permis une prise de décision rapide. La section 5 du rapport témoigne sur les choix qui ont été gardés et implémentés au niveau de design d'application.

6.3 QUELQUES SCENARIOS DE TEST

7 Conclusion

7.1 DEFIS ET SOLUTIONS

7.2 LIMITES ET POSSIBILITES DE DEVELOPPEMENT

Les points forts et faiblesses d'application, le temps nécessaire pour développer le concept avec deux personnes, les sources consultées pour la partie pratique (i.e. les slides, ou techniques vues au TP, ainsi que les conseils d'assistante, mais aussi w3schools ou bien stackoverflow). Finalement quelques mots sur l'expérience du projet.