

Alyacoubi Reda

Brault Clément

Champion Antoine

Orain Alexandre

**Rapport de projet**

**Refonte du logiciel du foyer des élèves**

Table des matières

[État de l’art 2](#_Toc9948248)

[I – Organisation et gestion de projet 3](#_Toc9948249)

[II – Répartition des tâches et familiarisation avec les outils 7](#_Toc9948250)

[III – Module d’authentification 8](#_Toc9948251)

[IV – Pages d’authentification en ligne (Front-end) 8](#_Toc9948252)

[V – Intégration des pages d’authentification sur le serveur 10](#_Toc9948253)

[VI – Création de la base de données 11](#_Toc9948254)

[VII – Interface sur l’ordinateur du foyer des élèves 12](#_Toc9948255)

[VIII – Interface de paiement en ligne 14](#_Toc9948256)

[IX – Synchronisation interface foyer et site 15](#_Toc9948257)

[X – Première phase de test utilisateurs 18](#_Toc9948258)

[XIII – Objectifs de la seconde phase du projet 18](#_Toc9948259)

État de l’art

Le foyer des élèves est au centre de notre vie aux Arts et Métiers.

L’objectif de ce projet est d'améliorer sa gestion en effectuant une refonte du logiciel actuellement utilisé.

Actuellement, ce logiciel permet seulement de décompter les consommations des personnes sur leurs comptes, il permet également d’avoir un historique très sommaire des consommations et l’argent restant sur les comptes.

En l'état, il s'agit d'une interface front-end liée à quelques macros Access, le logiciel est très basique et ne permet pas d’avoir une gestion des stocks précise, ce qui est pénalisant pour la comptabilité.

L’idée serait donc de créer un logiciel qui permettrait une gestion des stocks et des approvisionnements automatisée et poussée, mais aussi avec une interface web adaptée aux ordinateurs et aux smartphones avec la possibilité pour un utilisateur de recharger son compte et de consulter son historique, et pour les administrateurs, avoir accès à un panneau de contrôle complet.

I – Organisation et gestion de projet

Afin de cadrer le projet et de ne pas partir en aveugle, nous avons utilisé plusieurs séances afin de définir la portée du projet à l’aide de différents outils.

Notre premier objectif était de définir la portée du projet, les évolutions par rapport au logiciel précédent et de mettre en lumière des liens entre les idées des différents membres. Nous avons donc réalisé une Mind Map (carte des idées).

Dans la continuité de cette Mind Map, nous avons réalisé une liste exhaustive des fonctions que notre plateforme allait offrir. En parallèle, nous avons soumis nos idées à la promotion et réalisé des sondages afin d’avoir le plus de retours possibles.

Nous leur avons soumis deux questions, et voici les réponses qui revenaient le plus souvent :

**Q :** Que faudrait-il changer dans le système actuel ?

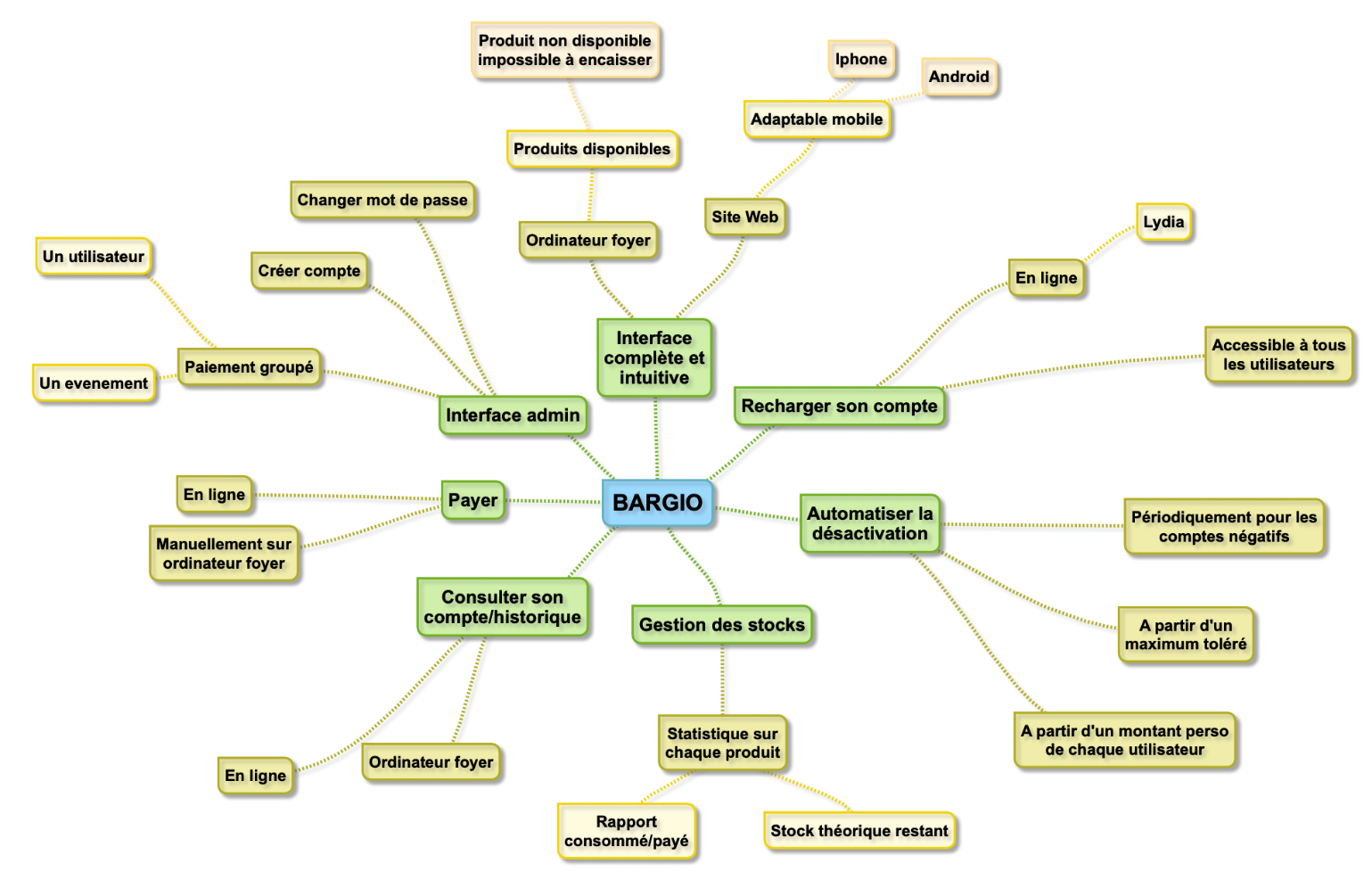
**R :** Davantage de précisions sur l’historique personnel de consommations (type de produit, quantité, date)

Recharge du compte autonome et en ligne

Changement du design de l’interface utilisateur

Fusion avec compte Borgia

Protection du compte par mot de passe personnel



*Fig 1. Mind Map*

**Q :** Quelle(s) fonctionnalité(s) supplémentaire(s) voudriez-vous ajouter dans ce programme ?

**R :** Déblocage du compte automatique suite au rechargement

Afficher des statistiques (grands consommateurs, records journaliers...)

Recevoir une notification sur téléphone portable lors d’une consommation

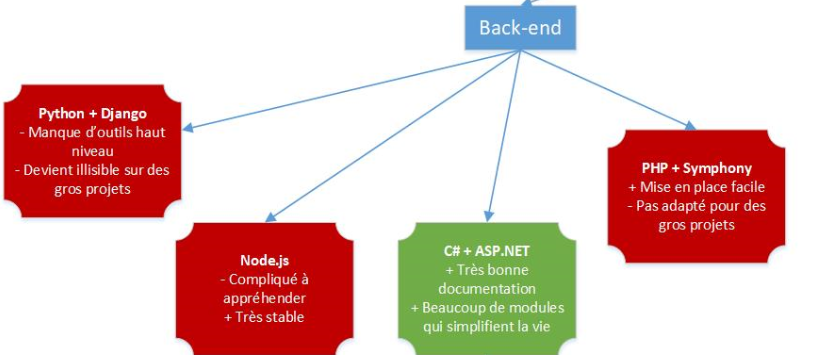
Seuil de blocage de compte personnalisable

Pouvoir annuler une consommation dûe à une faute de frappe

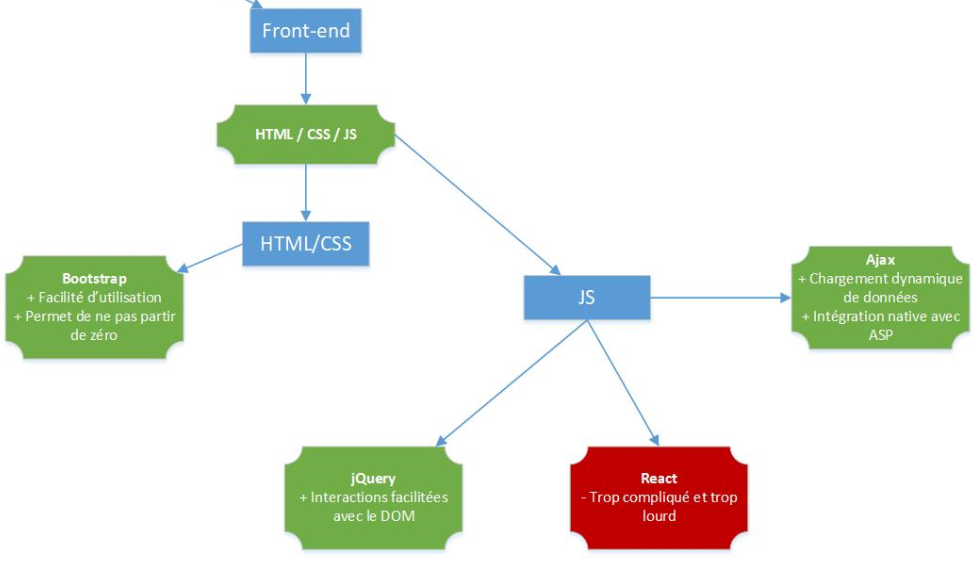
Une fois terminée, cette liste des fonctionnalités a pu être exploitée afin d’entamer une réflexion sur quelles technologies utiliser.

Une plateforme web se sépare en deux parties : le front-end qui concerne tout ce qui est exécuté côté client (design, layout, scripting…), et le back-end qui est exécuté côté serveur (gestion du hardware, bases de données, sécurité…).

A la fois pour le front et le back-end, il existe des *frameworks*, qui sont des infrastructures logicielles spécialisées permettant de créer les fondations d’autres projet. Nous avons dû étudier et comparer différents frameworks afin de trouver le plus pertinent. Nous avons choisi ASP.NET pour le back-end (cf fig. 2), et l’association de bibliothèques Bootstrap/jQuery pour le front-end (cf fig. 3).



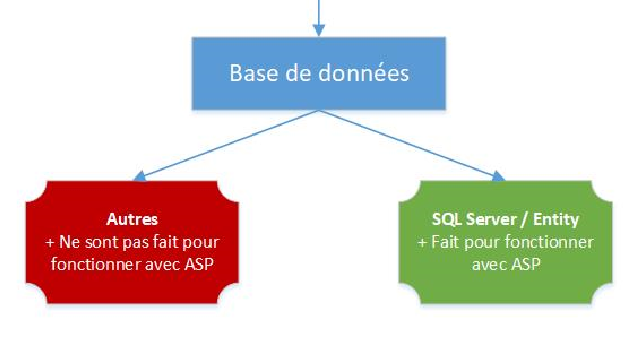
*Fig. 2 : Arbre des technologies (framework back-end)*



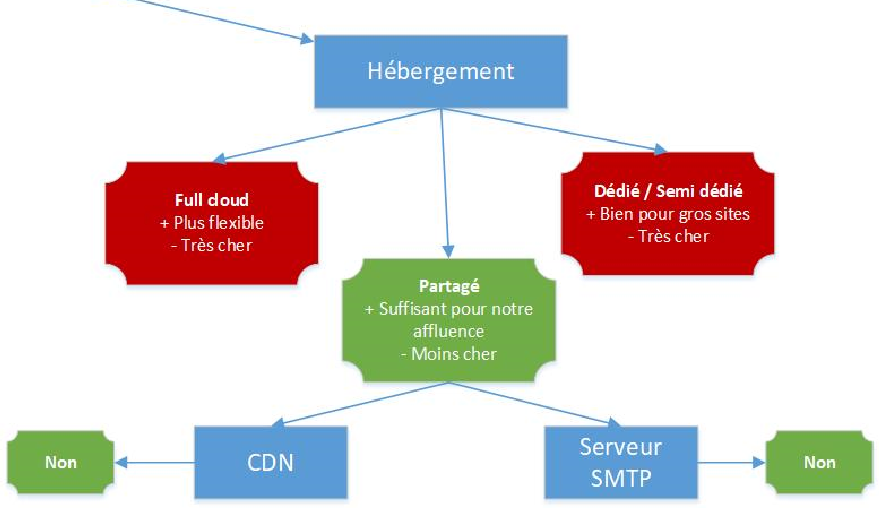
*Fig. 3 : Arbre des technologies (framework front-end)*

Une fois le framework déterminé, il a fallu choisir un *design-pattern*, qui représente une manière de programmer, d’organiser son code et de le hiérarchiser. Effectivement, notre framework en proposait deux différents, nous avons choisi Razor Pages (cf fig. 4).

Ces choix étant effectués, il ne nous restait plus qu’à choisir un système de bases de données, ainsi qu’un modèle d’hébergement. Nous sommes partis sur SQL Server couplé à son framework Entity, et un hébergement partagé (cf fig. 4,5).



*Fig. 4 : Arbre des technologies (bases de données)*

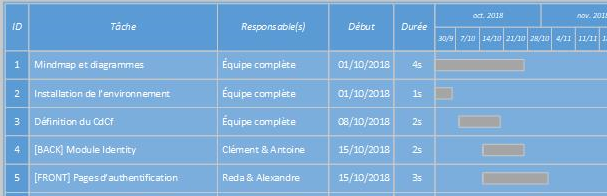


*Fig. 5 : Arbre des technologies (hébergement)*

II – Répartition des tâches et familiarisation avec les outils

A ce stade, nous avions une liste exhaustive des fonctionnalités à implémenter ainsi qu’un arbre des technologies complet.

La prochaine étape a donc été de réaliser un diagramme Gantt afin de relier les fonctionnalités aux technologies et de planifier le travail (cf fig. 6).



*Fig. 6 : Vue partielle du diagramme Gantt*

Le diagramme étant bien fourni (25 tâches sur 36 semaines), nous ne vous proposons qu’une vue partielle. Le diagramme entier est disponible en annexes.

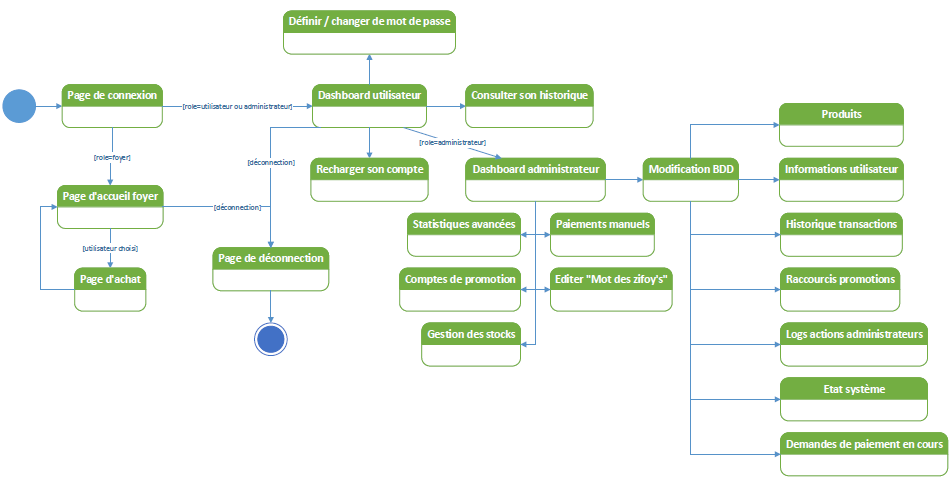
Lors de la réalisation de ce Gantt, nous avons constaté qu’il était compliqué de paralléliser différentes tâches : le back-end a besoin que le front-end soit au même avancement que lui pour pouvoir en réaliser une implémentation. Cela nous a amené à une logique de travail en 2 équipes, la première s’occupant exclusivement du front-end (MM. Alyacoubi et Orain), et la seconde exclusivement du back-end (MM. Brault et Champion). Les étapes intermédiaires telles que la tenue de projet, les phases de test et le passage en production sont cependant gérées par l’équipe entière.

Afin de réaliser ce diagramme, nous avons dû beaucoup nous documenter afin d’anticiper au maximum les différentes étapes du projet, et de savoir comment les appréhender.

Tous les membres de l’équipe n’ayant pas forcément la même expérience avec les outils de programmation, un temps d’appréhension et de familiarisation avec les outils de programmation a également été mis en place, et nous avons commencé avec les modules les plus simples à réaliser, c’est-à-dire la création de comptes et les pages de connexion/déconnection.

Pour le back-end, nous avons codé en C# avec ASP.NET Core 2 et Javascript sur Visual Studio, et pour le front-end, en HTML+CSS et Javascript sur Adobe Brackets.

Suite à cette première période, nous avons réalisé une arborescence complète de notre site afin de faire apparaître clairement la navigation au sein de celui-ci (cf fig. 7).



*Fig. 7 : Arborescence du site*

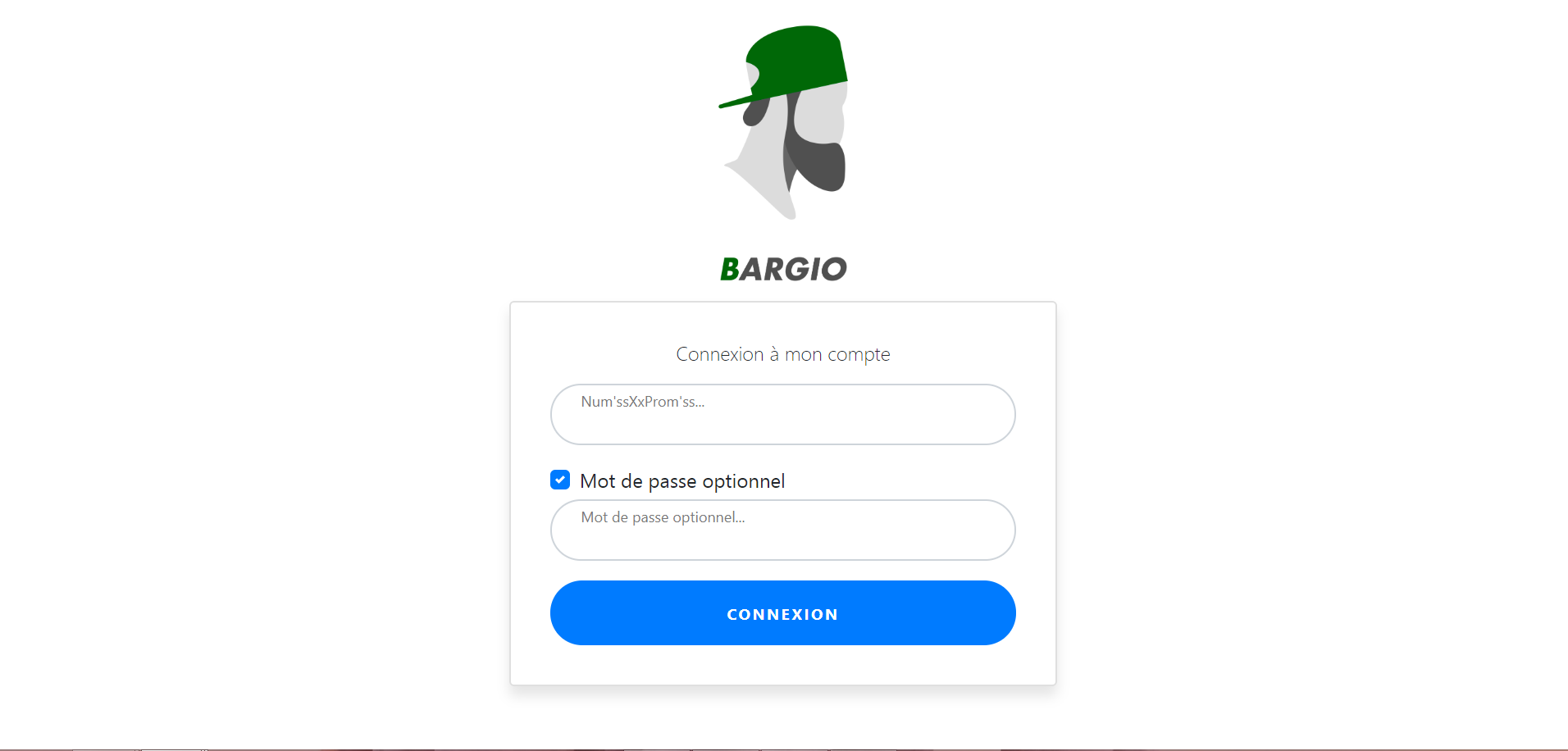
III – Module d’authentification

La première étape que nous avons réalisée concernait le module d’authentification : créer des bases de données d’utilisateurs qui peuvent se connecter à l’aide d’un mot de passe optionnel, gérer ce mot de passe et se déconnecter.

IV – Pages d’authentification en ligne (Front-end)

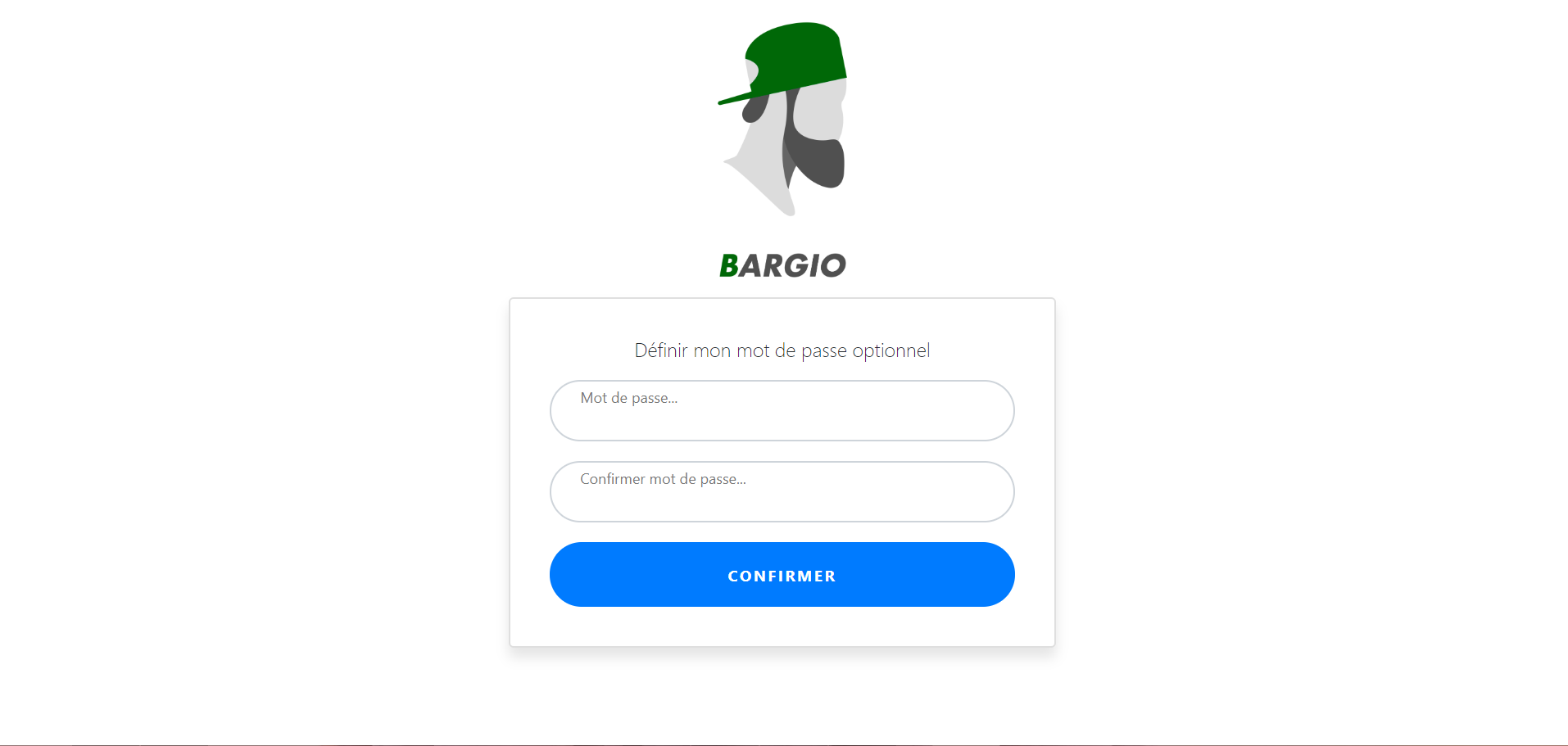
Après s’être familiarisé avec le langage choisi, nous nous sommes dans un premier temps intéressé à l’interface d’authentification utilisateur en ligne. Ces pages doivent avoir un format adaptable aux mobiles et aux ordinateurs.

La première page sur laquelle nous avons travaillé est la page d’accueil en ligne qui permet aux utilisateurs ou aux administrateurs de se connecter à leur Dashboard depuis leur mobile ou leur ordinateur personnel (Fig. 8). L’authentification de l’utilisateur nécessite d’entrer un identifiant sous la forme Num’ssXxProm’ss, qui est unique à chaque étudiant Gadz’Art ainsi qu’un mot de passe, si celui-ci a préalablement été défini. Ce mot de passe étant optionnel.

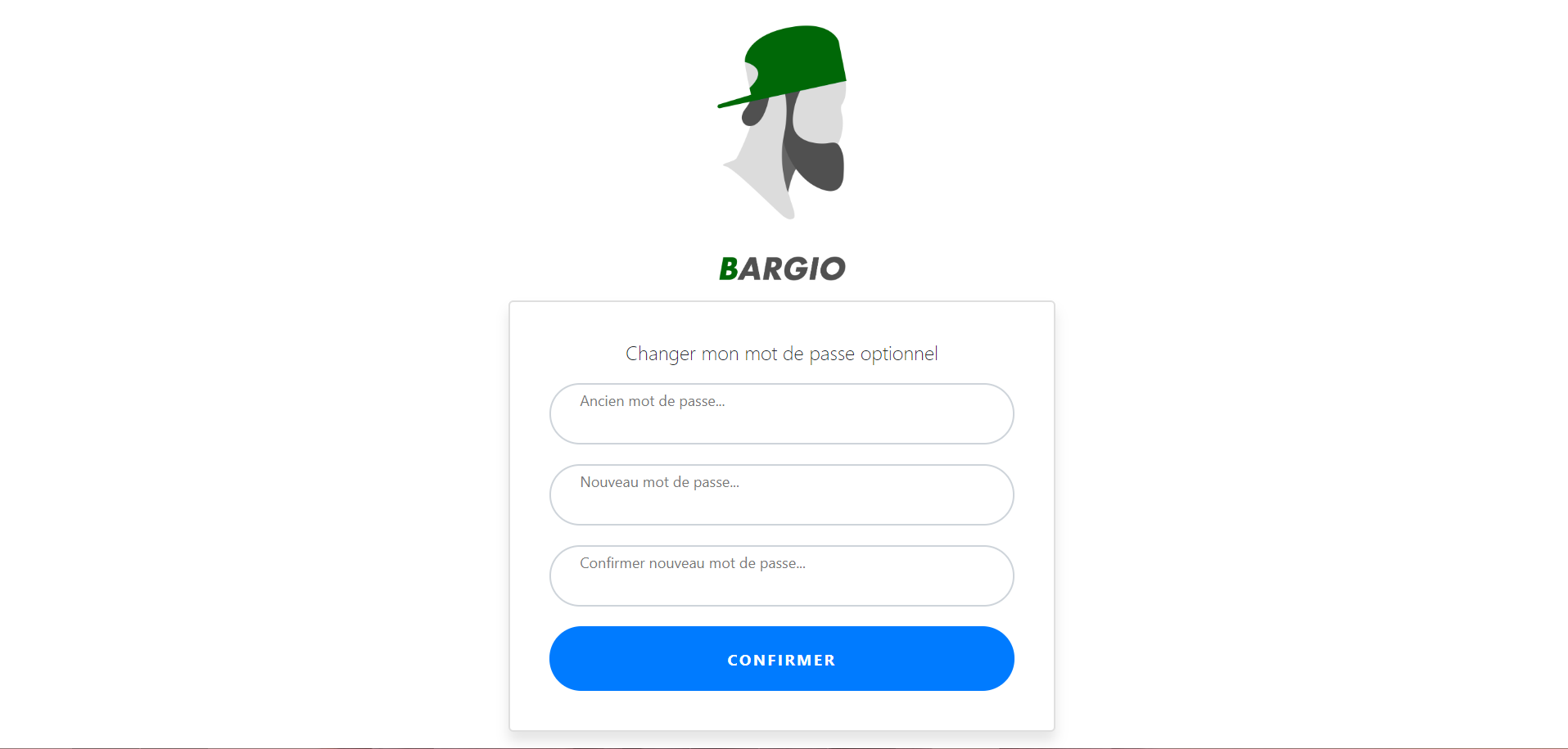


*Fig. 8 : Page d’accueil en ligne*

Nous avons ensuite codé les pages de création de mot de passe ainsi que celle de changement de mot de passe (Fig.9 et Fig. 10).



*Fig. 9 : Page de création du mot de passe optionnel*



*Fig. 10 : Page de changement du mot de passe*

Bien que ces pages soient visuellement simples et peu nombreuses, ce résultat a été difficile à obtenir. En effet, il s’agissait des premières pages que nous réalisions entièrement et concrètement : il était parfois difficile d’identifier nos erreurs ou de réaliser le codage permettant d’obtenir le rendu que l’on souhaitait. De plus, il fallait choisir et concevoir un design simple et soigné que l’on retrouverait sur toutes les futurs pages qui composeront la plateforme de Bargio.

V – Intégration des pages d’authentification sur le serveur

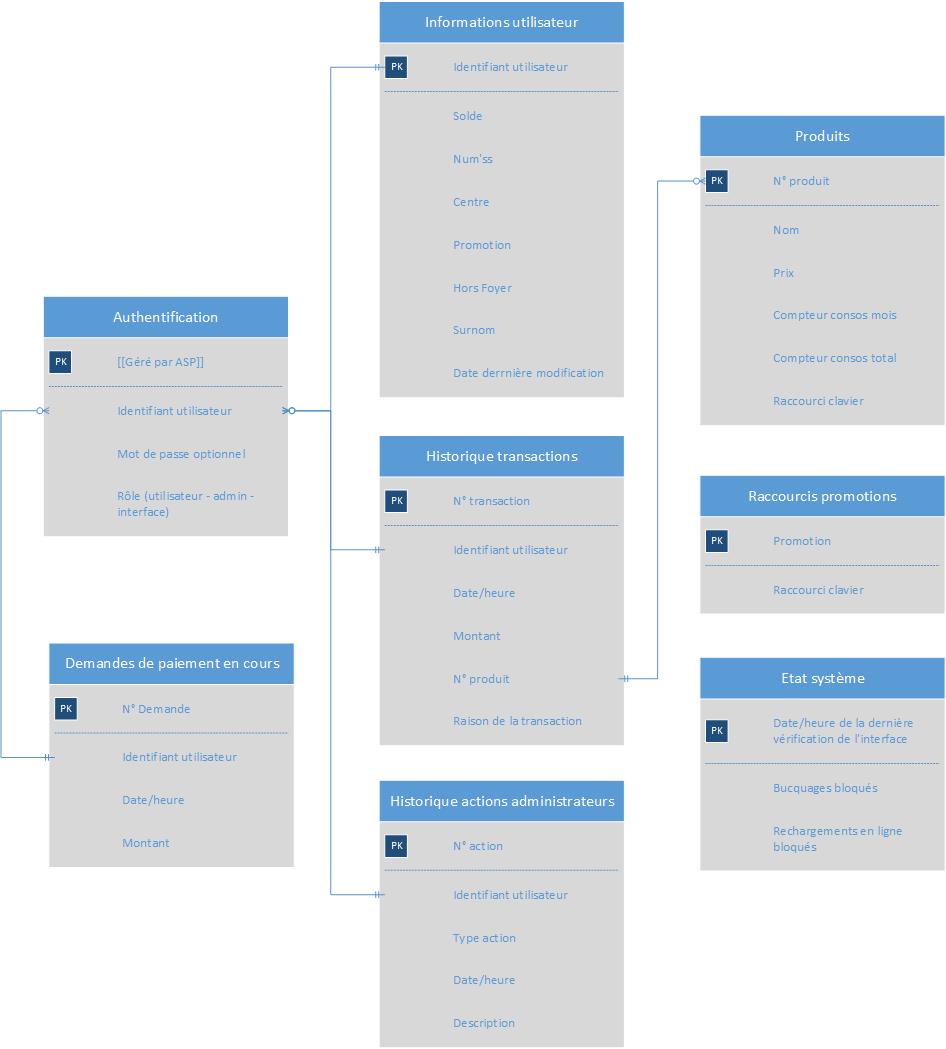
ASP.NET propose un système d’authentification intégré appelé Identity. Ce module contient un type générique UserManager<IdentityUser> qu’il faut appeler par injection de dépendances. Cette classe permet déjà de :

* Créer un nouvel utilisateur avec mot de passe
* Changer le mot de passe d’un utilisateur
* Faire se connecter et se déconnecter un utilisateur

La classe IdentityUser ne permet cependant pas de désactiver le mot de passe par défaut, nous avons alors dû créer une classe dérivée.

VI – Création de la base de données

Nous avons ensuite créé le schéma de notre base de données en fonction de nos besoins :



*Fig. 11 : Schéma de la base de données*

Pour ce faire, nous avons utilisé le moteur de base de données SQL Server intégré à ASP.NET ainsi que sa surcouche d’abstraction Entity Core. La philosophie d’Entity est très différente des moteurs SQL usuels : il n’existe pas de tables mais des « entités », pouvant stocker des variables classiques ainsi que des objets C#, mais aussi des données non-structurées à la manière de NoSQL. Les requêtes s’effectuent via LINQ.

Par exemple, si l’on désire récupérer le total des transactions effectuées entre une date A et une date B, nous effectuons la requête suivante :

var total = ApplicationDbContext.HistoriqueTransactions

.Where(o => o.Date > A && o.Date < B)

.Select(o => o.Montant)

.ToList()

.Sum();

VII – Interface sur l’ordinateur du foyer des élèves

Contrairement à la plateforme en ligne qui sera une nouveauté, une interface d’achats existe déjà sur l’ordinateur au sein du foyer des élèves. Un des objectifs définis pour notre projet était de refaire cette interface, afin de réduire les bugs et d’améliorer le visuel et l’ergonomie. Cette interface permet la connexion des utilisateurs et des administrateurs ainsi que l’achat de consommations, et elle reste en permanence ouverte sur l’ordinateur du foyer.

Les éléments présents avant la connexion d’un utilisateur sont les suivants (visible sur la Fig. 12) :

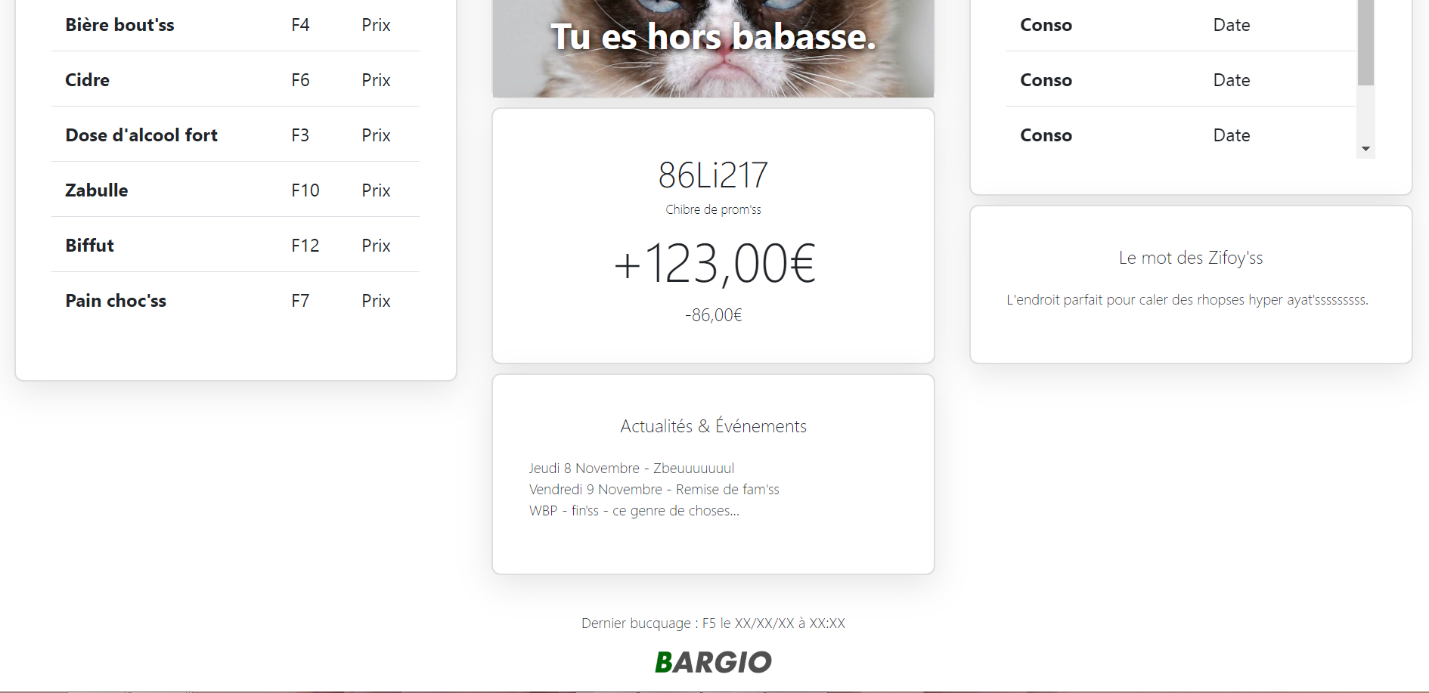
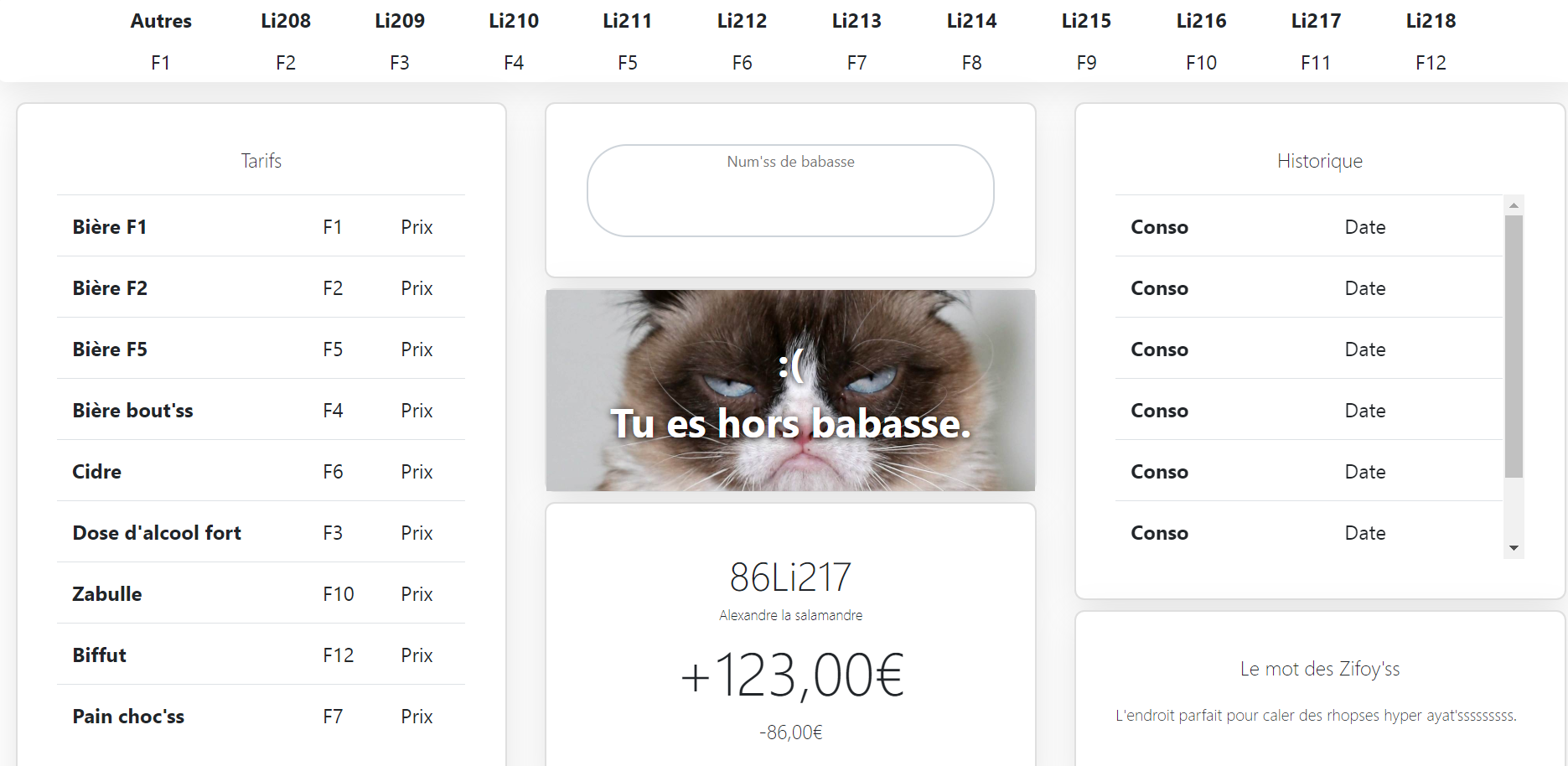
* Espace de renseignement du code de connexion de l’utilisateur (Num’ss de babasse)
* Tarifs des produits disponibles
* Une zone d’information : « Actualités & Évènements »
* Une zone dédiée au message à transmettre de la part des gérants du foyer : « Le mot des Zifoy’ss »
* Les raccourcis des différentes promotions
* Achat réalisé par le dernier utilisateur : « Dernier buquage »

Le fonctionnement pour l’utilisateur est le suivant : l’utilisateur entre son code de connexion (qui est indépendant des identifiants de connexion en ligne), qui est par défaut son numéro de Gadz’Art, dans l’espace défini. Il effectue la connexion à son compte en renseignant sa promotion au sein de l’école à l’aide du clavier (par exemple la touche F11 pour la promotion Li217). En renseignant sa promotion, l’utilisateur reste sur la même page. Des informations supplémentaires apparaissent :

* Si son solde n’est pas suffisant (généralement lorsqu’il est négatif) pour pouvoir utiliser son compte, une photo apparait annonçant : « Tu es hors babasse » ainsi que le montant de son solde. L’utilisateur doit donc recharger son compte afin de procéder à un achat.
* Si son solde est suffisant, il accède à son solde, son historique et son encours d’achat. Il peut alors procéder à des achats. Pour cela, il sélectionne les produits qu’il souhaite consommer à l’aide de la souris sur la zone des tarifs, ou bien en utilisant les raccourcis sur le clavier. L’encours augmente sans modifier le solde pour le moment. L’utilisateur appuie ensuite sur la touche *Entrée* afin de valider ses achats.

Malgré la demande de certains élèves, nous avons décidé de ne pas afficher sur cette interface les statistiques de consommation de l’utilisateur. Nous avons pensé qu’il s’agirait d’un moyen de se défier entre élèves et donc de pousser certains à la surconsommation dans le seul but d’avoir des « bonnes » statistiques, ce qui peut être dangereux. De plus, cela relève du privé.

Les éléments présents sur cette page restent les mêmes que ceux actuellement présents. Nous avons amélioré l’esthétique et la visibilité des informations.



*Fig. 12 : Interface du foyer*

Nous n’avons pour le moment réalisé que le front-end, l’implémentation sur le serveur reste à définir au second semestre.

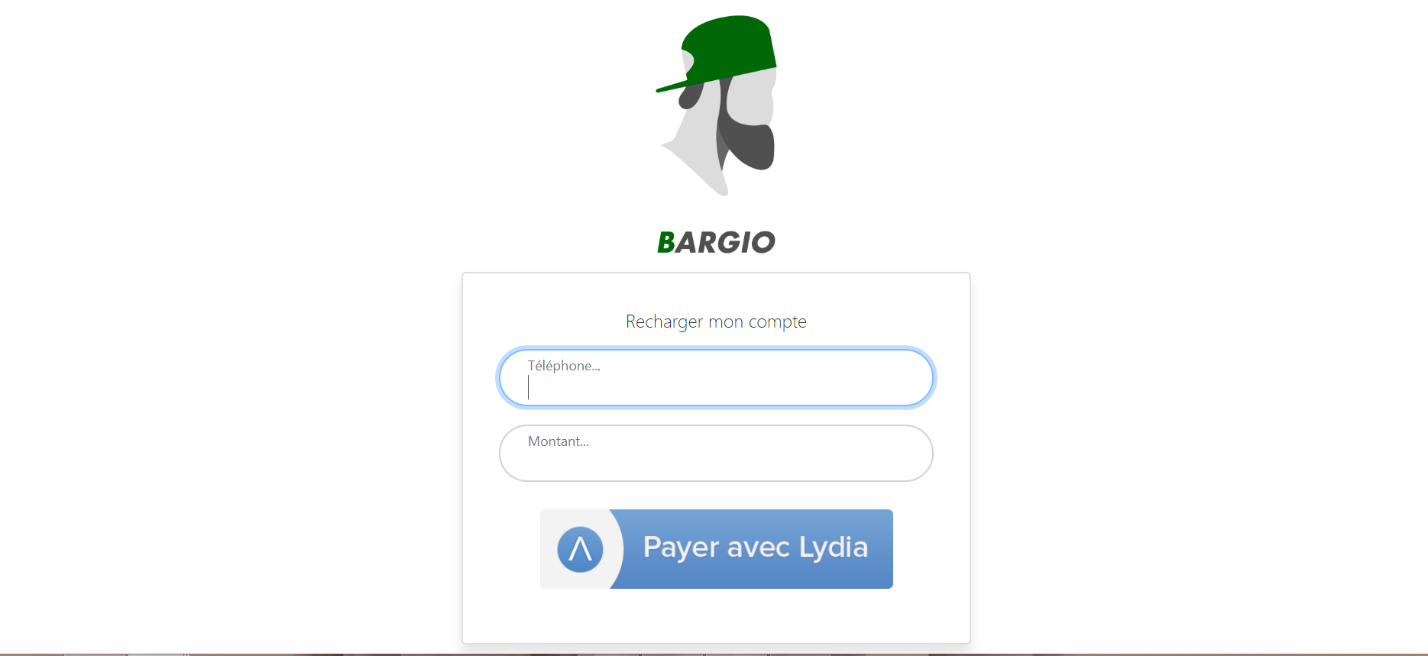
VIII – Interface de paiement en ligne

Le but premier de notre projet était de rendre possible le rechargement de son compte en ligne. D’une part, le rechargement est plus facile pour l’utilisateur et plus rapide, d’une autre part, il décharge les responsables du foyer de recharger les comptes à chaque fois qu’un utilisateur le demande. En effet, jusque maintenant, l’utilisateur qui voulait recharger son compte devait aller voir un responsable et lui effectuer un virement. Ce dernier devait aller sur l’ordinateur du foyer pour renseigner le montant payé par l’utilisateur.

Nous avons donc réalisé une page de rechargement du compte du foyer (Fig. 13). Le rechargement se fait via Lydia, qui est une société spécialisée dans le paiement en ligne. Pour disposer d’un compte il suffit d’un numéro de téléphone qui est donc unique pour chaque compte Lydia. Notre page en question demande à l’utilisateur son numéro de téléphone (qui n’est pas stocké, simplement transmis à Lydia).

En back-end, un jeton unique de transaction est créé et stocké (dans l’entité Demandes de paiement), puis transmis à Lydia via son API, accompagné du montant et du numéro de téléphone. Cette API attend que l’utilisateur valide le paiement, puis envoie une requête POST sur une de nos pages avec le jeton de transaction et l’état du paiement (finalisé ou annulé).

Nous pouvons alors recharger l’utilisateur du montant qu’il avait renseigné.



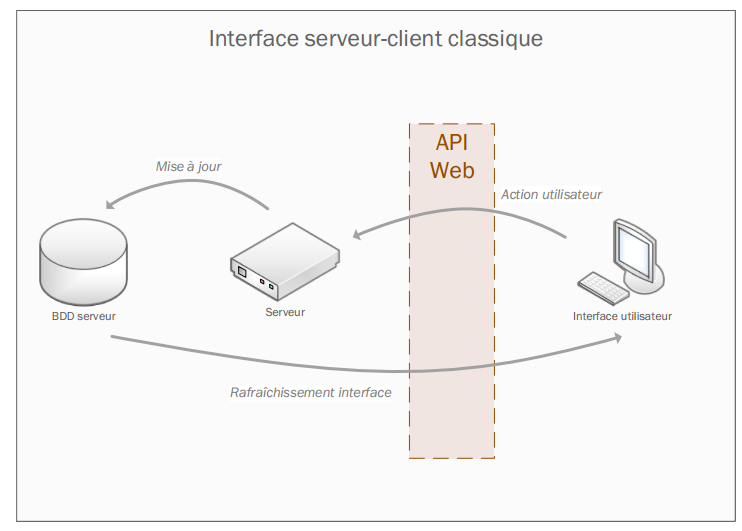
*Fig. 13 : Page de rechargement en ligne*

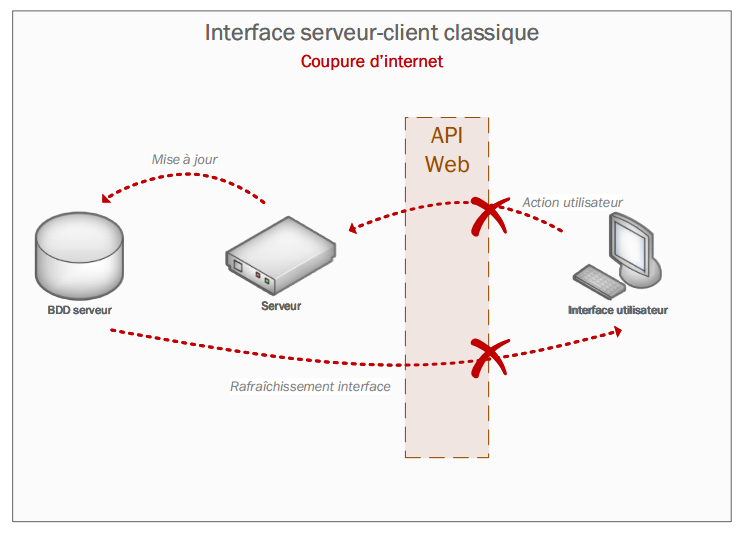
IX – Synchronisation interface foyer et site

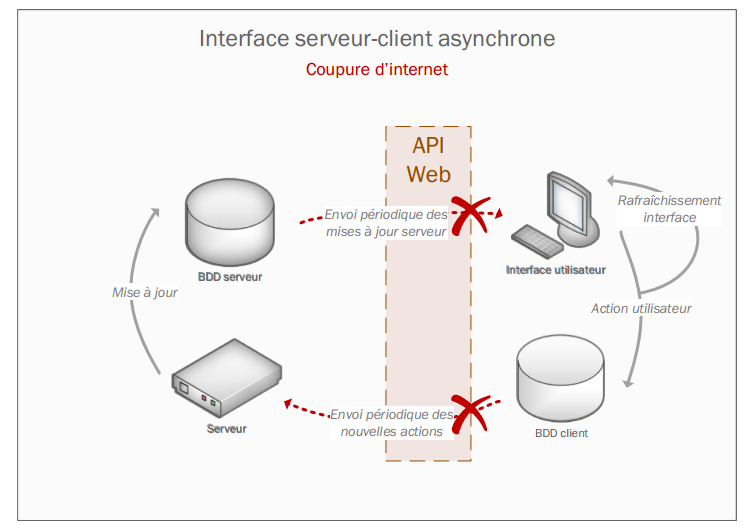
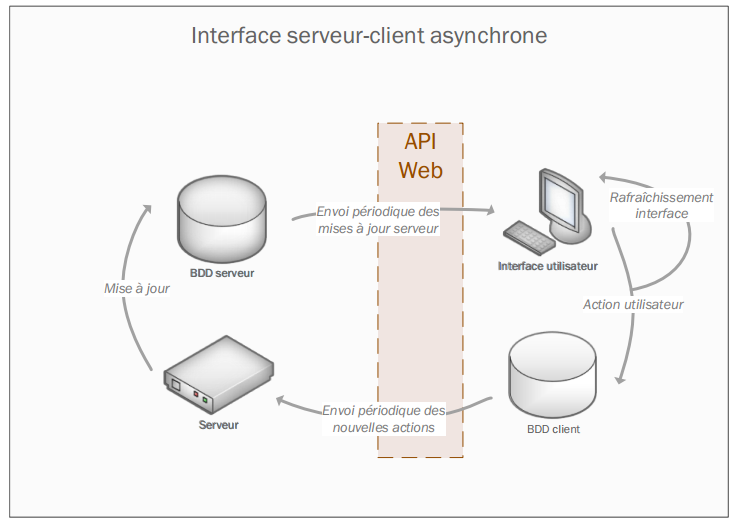
Une des plus grosses contraintes du projet est de faire en sorte que les gens puissent acheter des consommations même quand il n’y a pas internet sur l’ordinateur du foyer. Cela nous a amené à concevoir un système de synchronisation entre la page d’achat qui reste en permanence allumée sur l’ordinateur du foyer, et nos serveurs (cf fig. 14).

De cette manière, en créant une base de données côté client, même si il y a une coupure d’internet au foyer, les personnes pourront quand même payer leurs consommations et le tout sera resynchronisé dès qu’internet sera revenu.

Le code de cette interface sera effectué au second semestre, nous utiliserons du javascript asynchrone avec Dexie.js comme moteur de base de données locale.







*Fig 14. : Interface asynchrone pour la page d’achat*

X – Première phase de test utilisateurs

Nous avons rassemblé des volontaires devant la page d’achat en leur demandant des actions simples afin de vérifier que l’ergonomie était bonne, puis devant la navigation utilisateur (connexion, gestion de mot de passe et rechargement), afin de vérifier cette fois à la fois l’ergonomie et les potentiels bugs.

Les deux tests ont été tout à fait concluant, les utilisateurs ayant trouvé le site pratique à utiliser, et seulement des bugs mineurs ont été détectés (problèmes d’affichage sur certains navigateurs…), qui ont été depuis corrigés.

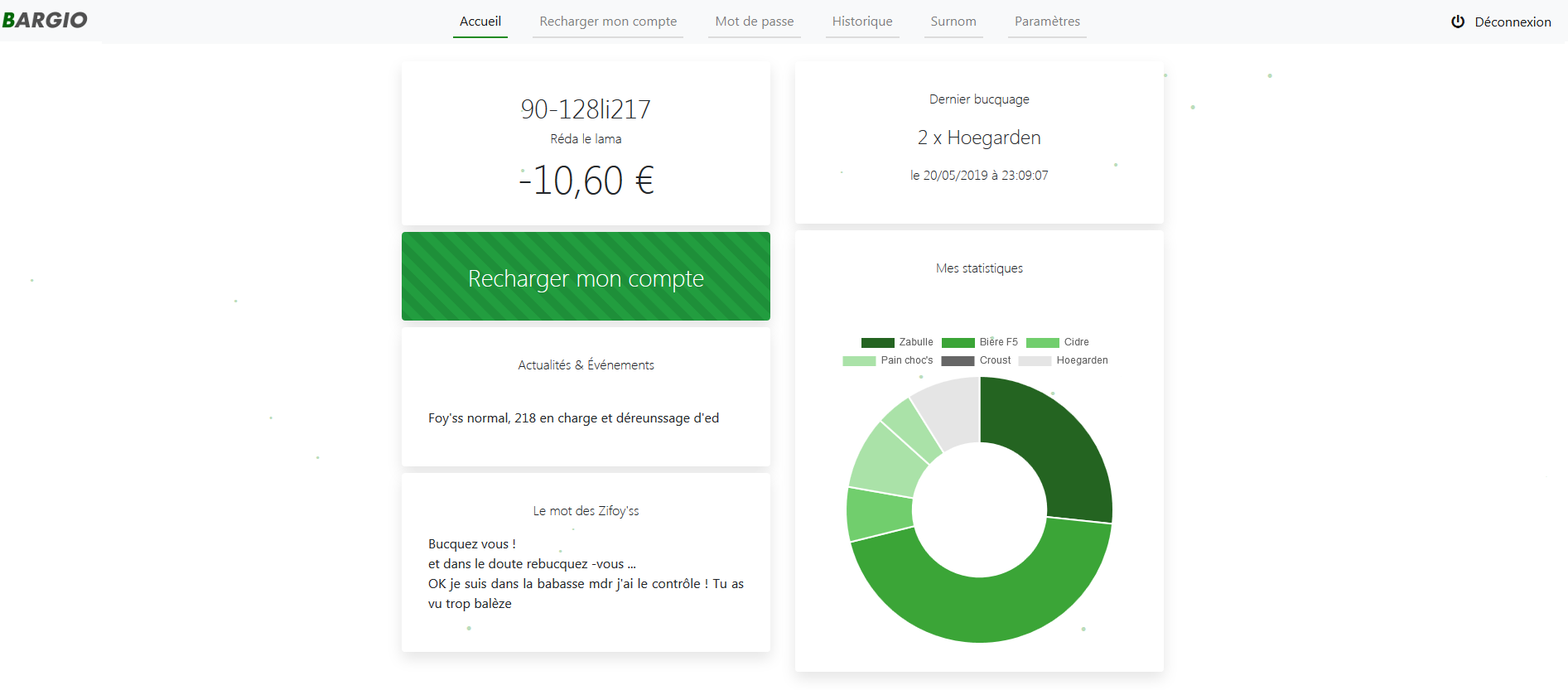
XIII – Dashboard utilisateur

Après avoir réalisé la page de connexion en ligne des utilisateurs, qui permet d’ailleurs également la connexion des administrateurs, le Dashboard utilisateur devait être réalisé, c’est-à-dire la page d’accueil des utilisateurs une fois connectés. Elles sont codées de la même façon que précédemment.

On tenait à ce que cette page d’accueil affiche les informations importantes liés à un compte :

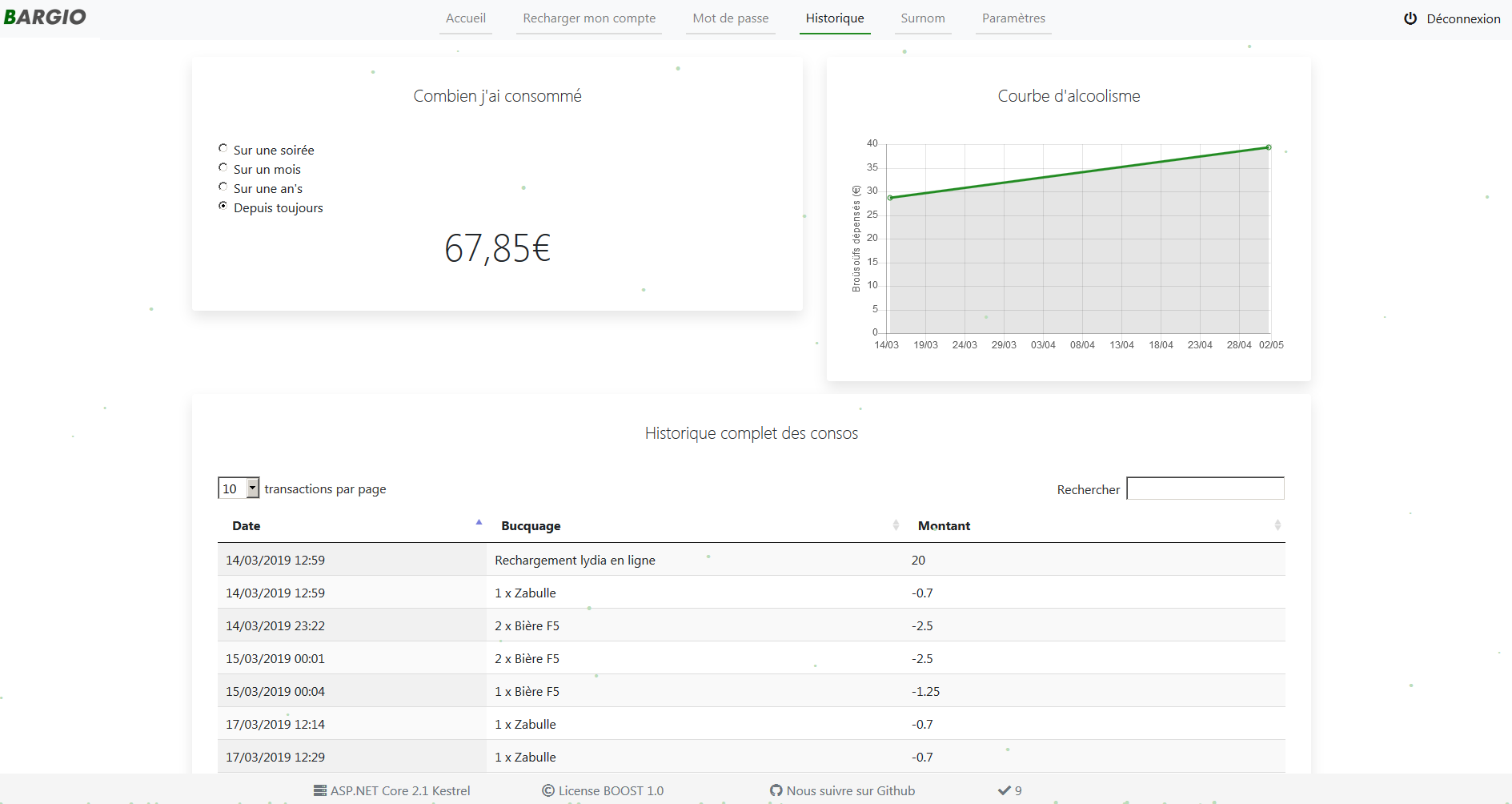
* Le crédit sur le compte
* Les statistiques générales
* Les informations à transmettre des gérants du foyer
* L’actualité et les évènements à venir
* La dernière consommation réalisée (qui permet de vérifier si quelqu’un a usurpé le compte récemment)
* Un bouton de rechargement qui permet d’accéder à la page de la figure 13

Sur un ordinateur, cette page ressemble à la photo ci-dessous :

  
*Fig 15. : Interface d’accueil des utilisateurs*

Cette page permet également d’accéder aux différentes pages du site :

* La zone de modification du mot de passe visualisable en figure 10
* La page de rechargement mobile visualisable en figure 13
* Les paramètres qui permettent de verrouiller son compte (empêcher le payement via son compte sur l’interface de payement du foyer)
* La page de modification du surnom (affiché sur l’ordinateur du foyer)
* L’historique des consommations, qui permet notamment de voir ses consommations sur une soirée précise, un mois, une année ou depuis la création de son compte. Cette page est visible sur la figure suivante.



*Fig 16. : Page de l’historique de consommation utilisateur*

Petit blabla sur la BDD utilisateur

XIV – Dashboard administrateur

Le Dashboard administrateur a ensuite été réalisé et est accessible via la page de connexion (figure 8). La volonté des gérants du foyer était d’avoir une meilleure visibilité sur les consommations et les produits, ainsi que de facilité les verrouillages de compte, la gestion des anciens élèves voulant consommer et les paiements automatiques de plusieurs personnes.

* Page d’accueil : Cette page permet dans un premier temps d’accéder aux autres pages qui seront décrites ci-après. Elle permet également d’afficher les derniers payements sur effectué sur l’ordinateur du foyer, permettant de vérifier si un élève sortant du foyer avec une consommation a bien validé son payement (et donc ne vole pas).
* Historique : il permet de répertorier l’historique des achats ainsi que des consommations. Il permet ainsi d’optimiser les futures commandes. Il permettra aussi, on l’espère d’améliorer les estimations lors des évènements importants de l’année.
* Payements groupés : Cette page permet de sélectionner un groupe d’élèves et de créditer ou de prélever un montant sur leurs comptes. Cela devaient être réalisé manuellement et pouvait être fastidieux. Cela est utile lors d’évènements collectifs nécessitant le payement de chaque consommateur pour un service ou un produit.
* Mode archi : Le mode archi consiste à ne plus pouvoir être en négatif sur son compte, ce qui est aujourd’hui permit pour les promotions présentes à l’école. Cette page permet donc de mettre des promotions entières en mode archi.

Petit blabla sur la BDD admin

XV – Initialisation des serveurs et mise en place du logiciel

Pour héberger Bargio, il fallait un serveur Windows, puisqu’il a été développé à l’aide de technologie Windows, c’est-à-dire un moteur de base de données Microsoft SQL (MSSQL). Il s’agit d’un système de gestion de base de données (SGBD) en langage SQL incorporant entre autres un logiciel qui permet de stocker des infos dans une base de données., fonctionnant sur OS Windows ainsi que Mac OS.

Parmi les offres disponibles, nous avons décidé d’opter pour la solution la moins chère et qui avait Plesk, une interface de gestion de serveur extrêmement pratique. L’hébergeur s’appelle A2Hosting. Il s’agit d’un hébergement partagé avec d’autres sites pour que ça coûte moins cher. **Les coûts mensuels de Bargio sont donc de 8 euros**.

Petit blabla sur la mise en place ?

XVII – Retour utilisateur, maintenance et amélioration continue

Après la mise en ligne de Bargio, les utilisateurs (c’est-à-dire les deux promotions présentes à l’écoles) ont fait remonter les bugs qu’ils ont rencontrés lors de l’utilisation du site. Ces bugs, souvent anodins, ont pu ainsi être corrigé pendant que celui-ci continue à fonctionner. Il était intéressant de voir toutes ces erreurs que nous n’avions pas prévues ni même imaginés lors du codage de Bargio. Voici une liste non exhaustive des bugs corrigés qui n’ont pu être repéré que lors du lancement du site :

* Verrouillage automatique hebdomadaire des comptes en négatif non fonctionnel
* Affichage du surnom du précédent connecté
* Le rechargement mobile nécessite de mettre une virgule et non un point
* Longueur de surnom trop long, pas assez de stockage (mise en place d’un maximum de longueur)
* Compte toujours verrouillé après rechargement