Deguine Damien

Naasse Youness

Rapport de projet

Le Frigo connecté

EI4 section AGI
Année scolaire 2016-2017

Réalisé à l'ISTIA, École d'ingénieur de l'Université d'Angers

Remerciements

Certaines personnes nous ont contribuées directement ou indirectement au bon déroulement de notre projet. Nous remercions :

- Mehdi Lhommeau, notre tuteur, qui a proposé ce projet très intéressant et qui nous a guidé pendant nos travaux.

- Guillaume Charbonnier pour nous avoir enseigné la gestion de projet et pour nous avoir partager des tableurs Excel nous ayant permis de gérer correctement le déroulement et l'avancée du projet.

- L'ISTIA pour nous avoir fourni le matériel nécessaire et nous avoir permis de commander des pièces utiles au projet.

Sommaire

Remerciements **1**

Introduction 3

Rappel du cahier des charges 4
A. Descriptif 4
B. Objectifs 4

Préparation du projet 6
A. Répartition des tâches au sein du groupe 6
B. Création des sprints avec la méthode agile 7

Déroulement du projet **10**
A. Détails des 5 sprints 9
B. Version finale 177

Outils utilisés 18**8**
Le stockage des données 188
Structure du site et accès aux données stockées 19
L'acquisition des photos avec la Picamera 20

Améliorations possibles 23**2**

Problèmes rencontrés 24**3**

Introduction

Nous sommes deux étudiants en seconde année du cycle ingénieur de l'ISTIA. Pendant notre année scolaire, nous avons 80 heures pour mener un projet. Un tuteur nous guide dans l'avancée de notre projet. Les heures sont réparties sur 4 mois, à raison d'une journée par semaine.

Nous mettons en application nos connaissances, et découvrons d'autres technologies non vues en cours. Nous nous familiarisons avec la gestion de projet, laquelle nous risquons fortement de retrouver en entreprise. Ce rapport est le compte-rendu de ce projet.

Rappel du cahier des charges

A. Descriptif

L'objectif du projet est de développer un réfrigérateur connecté.

Il s'agit d'un système automatique d'inventaire du contenu d'un frigo. Le but étant de permettre aux utilisateurs de pouvoir visualiser le contenu de leur frigo à tout moment, même à distance, d'avoir des alertes sur les aliments dont la date de péremption est proche, des rappels sur des denrées à acheter, la génération automatique des listes des courses, des recettes à partir d'aliments présents, des conseils diététiques, etc..

B. Objectifs

La photographie de l'intérieur du frigidaire sera réalisée à l'aide d'une caméra du type PI CAM (Caméra pour Raspberry PI) reliée à une raspberry PI 3.

Cette carte embarquera, également, le serveur WEB et l'application WEB (responsive design).

A chaque ouverture de la porte, la caméra prendra une ou plusieurs photos des contenus du frigo. L'objectif de la caméra est de capturer des images avec les contraintes suivantes :

— meilleure prise de vue possible de l'ensemble des aliments présents,

— éviter les prises d'images potentiellement occultées par la présence de l'utilisateur.

La présentation des images sur le serveur WEB devra permettre de naviguer dans l'historique des différentes prises de vue.

Voici les fonctionnalités attendues de l'application WEB :

— Accès au contenu de son frigo à la date courante.

— Visualisation du cliché.

— Possibilité de rajouter manuellement à la liste du contenu des aliments qui sont présents dans le frigo.

— Accès à un historique des contenus du frigo.

— Élaboration d’une liste de courses à partir :

— Du contenu actuel du frigo, du contenu à une date antérieure.

— De recettes proposées.

— D’anciennes listes de courses.

— Réception d’alertes si la date de péremption d’un aliment approche (la date de péremption sera saisie manuellement).

On peut imaginer, suivant l’avancement du projet, d’autres fonctionnalités comme la reconnaissance des aliments à partir de la photo (http://fr.openfoodfacts.org/).

Préparation du projet

A. Répartition des tâches au sein du groupe

Après avoir établi la liste précise des objectifs à partir du cahier des charges ci dessus, nous avons extrait trois grands axes de travail autour des trois composants principaux :

- La carte Raspberry PI (Binôme)
- L'application Web (Damien)
- La caméra PI (Younes)

Voici le catalogue final des tâches que nous nous sommes fixées à réaliser :

Backlog des tâches à effectuer :			
	Priorité	Estimation - PF	Sprint
Démarrage Raspberry PI			
Initialisation Raspberry PI	P1	3	S1
Choix d'un serveur web	P1	1	S1
Installation du serveur web	P1	5	S1
Tests de connexion au serveur	P2	3	S1
Application Web			
Choix du framework javascript	P2	2	S1
Préparation de l'environnement du travail	P2	2	S2
Création d'une base de données	P1	3	S2
Accès au contenu temps réel du frigo	P1	5	S2
Accès à l'historique des contenus du frigo	P1	5	S2
Ajout et retrait des aliments de la liste du contenu	P1	8	S3
Élaboration auto d'une liste de courses via archives	P1	5	S3
Ajout de recettes à la base de données	P1	5	S4
Élaboration auto d'une liste de courses via recettes stockées	P1	5	S4
Réception d'alertes si la date de péremption d'un aliment approche	P1	5	S3
Intégration des éléments sur la carte Raspberry Pi 3	P1	13	S5
Design du site web	P1	13	S5
Caméra			
Intégration de la caméra sur la carte	P1	5	S3
Réalisation d'un programme test de fonctionnement de la caméra	P2	8	S3
Appel du script caméra via site web test	P1	8	S4
Intégration du script sur l'application web	P1	13	S5
Reconnaissance des aliments à partir de la photo	P4	21	-
Total		138	

B. Création des sprints avec la méthode agile

Pour mener à bien ce projet, nous avons décidé d'appliquer la méthode SCRUM étudiée dans le cours de Gestion de Projet.

Nous avons choisi de découper notre temps de projet en 5 sprints, correspondant environ à des périodes de deux semaines chacun :

Sprint 1	
Début	7/12/2016
Fin	11/01/2017
PF réalisés à la date de fin	14

Sprint 2	
Début	11/1/2017
Fin	15/02/2017
PF réalisés à la date de fin	15

Sprint 3	
Début	01/03/2017
Fin	08/03/2017
PF réalisés à la date de fin	31

Sprint 4	
Début	15/03/2017
Fin	29/03/2017
PF réalisés à la date de fin	18

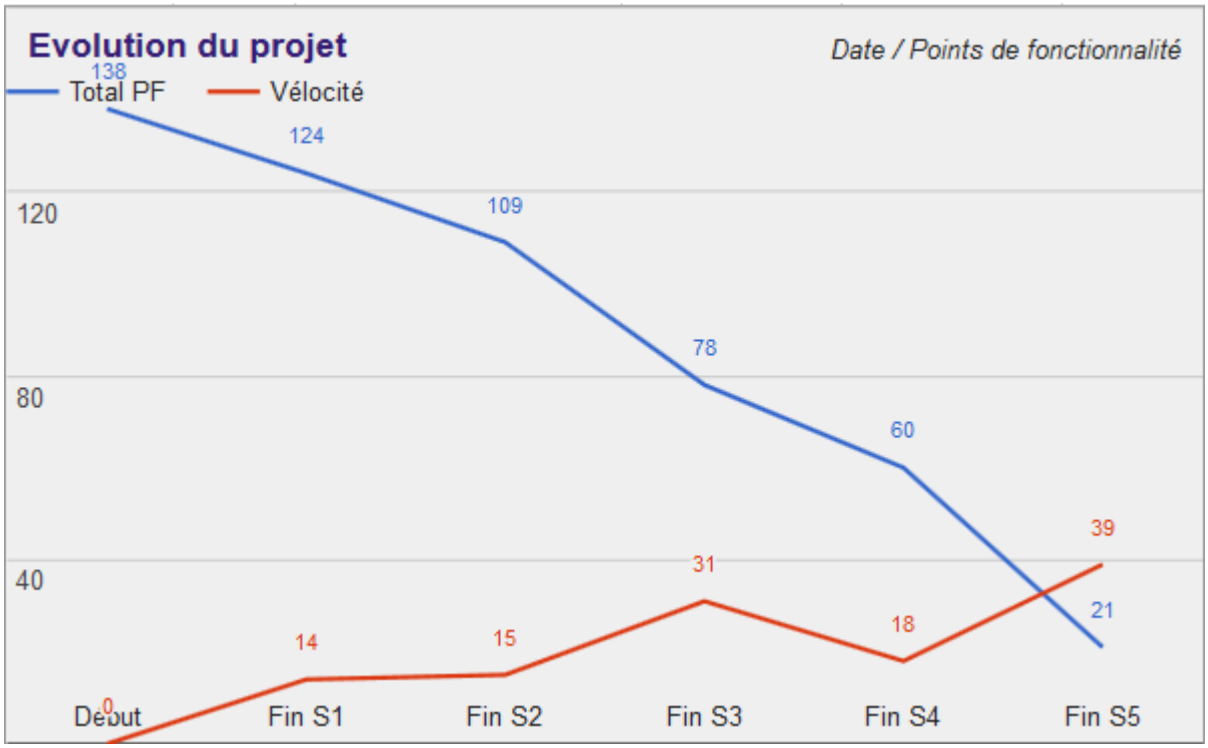
Sprint 5	
Début	05/04/2017
Fin	28/04/2017
PF réalisés à la date de fin	39

Après avoir établi le backlog des fonctionnalités, nous avons appliqué une priorité et une estimation de la difficulté à chacune de ces tâches, à partir des barèmes suivants :

Priorités	*1 - Maximale*	*2*	*3 - Moyenne*	*4*	*5 - Faible*

Difficulté	*1 - Très facile*	*2*	*3*	*5 - Moyen*	*8*	*13*	*21 - Difficile*

Voici le burndown chart du projet dans sa globalité :



Les 21 points de fonctionnalité restants viennent de la tâche bonus “Reconnaissance des aliments à partir de la photo” que nous n’avons pas eu le temps de mettre en place *(voir chap.5 - Amélioration envisagées).*

Déroulement du projet

A. Détails des 5 sprints

Sprint 1

Début	7/12/2016
Fin	11/01/2017

	Fonctionnalités concernées :	PF estimés :	Terminé le :
P1	Initialisation Raspberry PI	3	7/12
P1	Choix d'un serveur web	1	14/12
P1	Installation du serveur web	5	04/01/2017
P2	Tests de connexion au serveur	3	04/01/2017
P2	Choix du framework javascript	2	11/01/2017
P2	Préparation de l'environnement du travail	2	non terminé
	Total	16	14

Grille d'avancement :

Date de mélée	7/12/2016	14/12/2016	04/01/2017	11/01/2017
PF réalisés	3	1	8	2
PF restants	13	12	4	2

Le Sprint 1 a été le plus long, les quatre premières semaines de projet ayant été largement consacrées à l'étude du cahier des charges et à la réflexion, notamment concernant les outils que nous allions utiliser pour réaliser le travail demandé. Nous reviendrons sur ceux-ci plus loin, dans la section dédiée.

Nous avons doté notre Raspberry PI 3 du système d'exploitation Raspbian Jessie, dernière version en date du dérivé Debian optimisé pour la Raspberry. Ont été ensuite installé un serveur web Apache2 incluant une base de données mySQL, ainsi que Python nécessaire pour l'utilisation de la caméra.

Concernant l'application web, malgré de longues recherches pour intégrer un Framework Javascript, nous avons décidé de commencer avec du javascript natif afin d'avoir une base. NodeJS et AngularJS ont été envisagés.

Sprint 2

Début	10/2/2017
Fin	15/02/2017

	Fonctionnalités concernées :	PF estimés :	Terminé le :
P2	Préparation de l'environnement du travail	8	*10/2 midi*
P1	Création d'une base de données	3	*10/2 midi*
P1	Accès au contenu temps réel du frigo	5	*10/2 midi*
P2	Accès à l'historique des contenus du frigo	5	*10/2 soir*
P2	Ajout et retrait des aliments de la liste du co	8	-
	Total	29	21

Grille d'avancement :

Date de mélée	*10/2 midi*	*10/2 soir*	*15/2 midi*	*15/2 soir*
PF réalisés	16	5	0	0
PF restants	13	8	8	8

Le Sprint 2 a été entièrement consacré à la création du squelette de l'application web. Celui-ci fut réalisé sur un PC Windows, plus adéquat pour effectuer tests et simulations qu'un développement direct sur la Raspberry PI. Nous avions alors prévu d'intégrer l'application Web sur la carte lors du sprint 5, produisant ainsi le prototype final du projet.

La première fonctionnalité intégrée fut la connexion à la base de données mySQL, et l'affichage de contenu de la table "aliments" sur la page web. Ci-dessous, une capture d'écran du livrable Sprint 1 : une page HTML générée via un code PHP après que celui-ci se soit connecté à la base et ait récupéré les informations utiles :

- le Nom de l'aliment (non unique si date de péremption différente)
- la Quantité contenue dans le frigo
- la Date de Péremption

Nom de l'Aliment	Quantité	Date de Péremption
steak	8 U	2017-02-10
banane	2 U	2017-02-15
chocolat	4 G	2017-03-23
patate	18 U	2017-04-12
Beurre	500 G	2017-04-05
Beurre	200 G	2017-04-27
Bouteille de lait	1 L	2017-05-19
Carotte	2 U	2017-05-15

NomAliment	Quantite	DatePeremption	IDAliment	Type
steak	8	2017-02-10	1	U
banane	2	2017-02-15	3	U
chocolat	4	2017-03-23	6	G
patate	18	2017-04-12	7	U
Beurre	500	2017-04-05	10	G
Beurre	200	2017-04-27	13	G

Contenu de la table “Aliments”

```
//Code PHP d’appel BDD

$db = new PDO('mysql:host=localhost;dbname=frigo','admin','password');
$stmt = $db->prepare("SELECT NomAliment, Quantite, DatePeremption, Type FROM aliments");
$stmt->execute();

//Code PHP de création de table HTML utilisant “$stmt->fetch()”
```

Code PHP de la page “consult_frigo.php”

Contenu du frigo :

Nom de l'Aliment	Quantité	Date de Péremption
steak	8 U	2017-02-10
banane	2 U	2017-02-15
chocolat	4 G	2017-03-23
patate	18 U	2017-04-12
Beurre	500 G	2017-04-05
Beurre	200 G	2017-04-27

Affichage de la page sur le navigateur

Sprint 3

Début	1/3/2017
Fin	8/3/2017

	Fonctionnalités concernées :	PF estimés :	Terminé le :
P1	Ajout et retrait des aliments de la liste du contenu	8	1/3 soir
P1	Élaboration auto d'une liste de courses via archives	5	8/3 soir
P1	Réception d'alertes si la date de péremption d'un aliment approche	5	8/3 midi
P1	Intégration de la caméra sur la carte	5	8/3 soir
P2	Réalisation d'un programme test de fonctionnement de la caméra	8	8/3 soir
	Total	31	31

Grille d'avancement :

Date de mélée	1/3 midi	1/3 soir	8/3 midi	8/3 soir
PF réalisés	0	8	5	18
PF restants	31	23	18	0

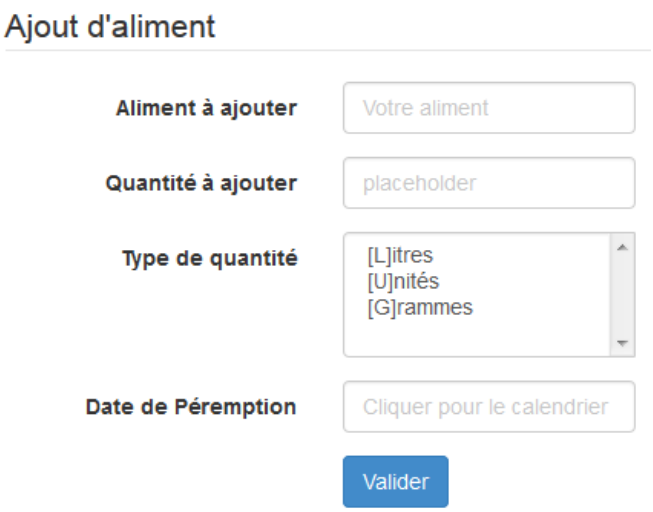
Au coeur du sprint 3, l'implémentation des fonctionnalités de modifications de la base de données sur l'application web, et celle de la caméra sur la raspberry PI, accompagnée de tests de fonctionnements.

La direction de l'application web, à ce moment du projet, était de créer une page PHP pour chacune des fonctionnalités afin d'avoir une meilleure vue d'ensemble de l'avancement.

La page "ajout_frigo.php" est composée du formulaire ci-dessous. Lors de la validation de celui-ci, un code PHP se déclenche. Celui-ci vérifie la complétion des champs de ce dernier et la cohérence des valeurs (date de péremption > date actuelle, quantité > 1) grâce aux variables POST obtenues, puis lance une des requêtes SQL suivantes :

- Si l'aliment et ses caractéristiques (date, type) existent déjà, on augmente juste la quantité avec UPDATE

- S'il n'existe aucune corrélation, on ajoute un nouvel élément dans la base de données avec INSERT INTO



Formulaire d'ajout dans la base

La page "retrait_frigo.php" intègre elle un autre formulaire, ne demandant que le nom d'un aliment et le nombre à enlever. Si l'aliment existe, suivant la quantité à supprimer demandée, une requête SQL "UPDATE" ou "DELETE" est envoyée :

Si suppression il y a, une deuxième requête SQL "INSERT INTO" est alors envoyée, celle-ci ajoutant à la table "archives" l'article supprimé.

Sprint 4

Début	15/03/2017
Fin	22/03/2017

	Fonctionnalités concernées :	PF estimés :	Terminé le :
P1	Ajout de recettes à la base de données	5	*29/3*
P1	Élaboration auto d'une liste de courses via recettes stockées	5	*22/3 midi*
P1	Design du site web	5	-
P1	Appel du script caméra via site web test	8	*29/3*
P1	Intégration du script sur l'application web	13	-
		Total 36	18

Grille d'avancement :

Date de mélée	*15/3 midi*	*15/3 soir*	*22/3*	*29/3*
PF réalisés	0	0	5	13
PF restants	36	36	31	18

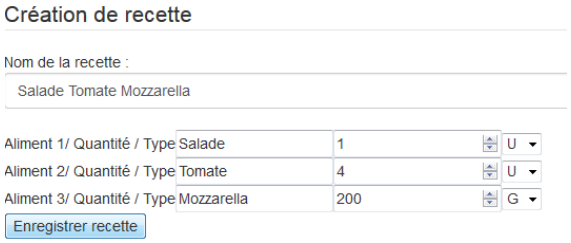
Lors du Sprint 4, nous avons ajouté une fonctionnalité concernant les recettes de cuisine. La page ainsi ajoutée “recettes_frigo.php” effectue une requête SQL récupérant les informations contenues dans la table recettes de la BDD.

Le champ ContenuRecette est de la forme : {Nom ingrédient 1;Quantité ingrédient 1;Type ingrédient 1;Nom ingrédient 2; Quantité ingrédient 2; Type ingrédient 2…}

ID	ContenuRecette	NomRecette
1	patate;10;U;steak;1;U	Steak Frites
2	tomate;3;U;fromage;50;G;jambon;3;U	Garniture Pizza Jambon-Fromage

Le contenu des recettes est alors mis en page dans un tableau à l’aide de la commande PHP “Split”, en se servant des points virgules précédents. La page possède également un formulaire permettant d’enregistrer une recette dans la base. L’utilisateur doit renseigner le nombre d’aliments, puis cliquer sur “Compléter la recette” afin de créer dynamiquement le nombre de champs correspondants. Après avoir donné un nom à la recette, il ne reste qu’à l’enregistrer en cliquant sur le bouton de validation.

Formulaire vierge



Formulaire après clic sur "Compléter"

Recette : Salade Tomate Mozzarella	
Salade	1U
Tomate	4U
Mozzarella	200G

Affichage de la table HTML après enregistrement

Concernant la partie Caméra :

Maintenant que notre programme marche et que notre caméra fonctionne bien, nous avons créé une simple page PHP qui aura comme rôle d'appeler notre script à partir d'un navigateur web. La tâche semble simple mais son développement sous raspbian et le serveur apache la rend un peu compliqué. Sous Unix, nous ne sommes jamais à l'abri des erreurs liées aux droit de lecture et d'écriture ainsi que quelques problèmes de compatibilités.

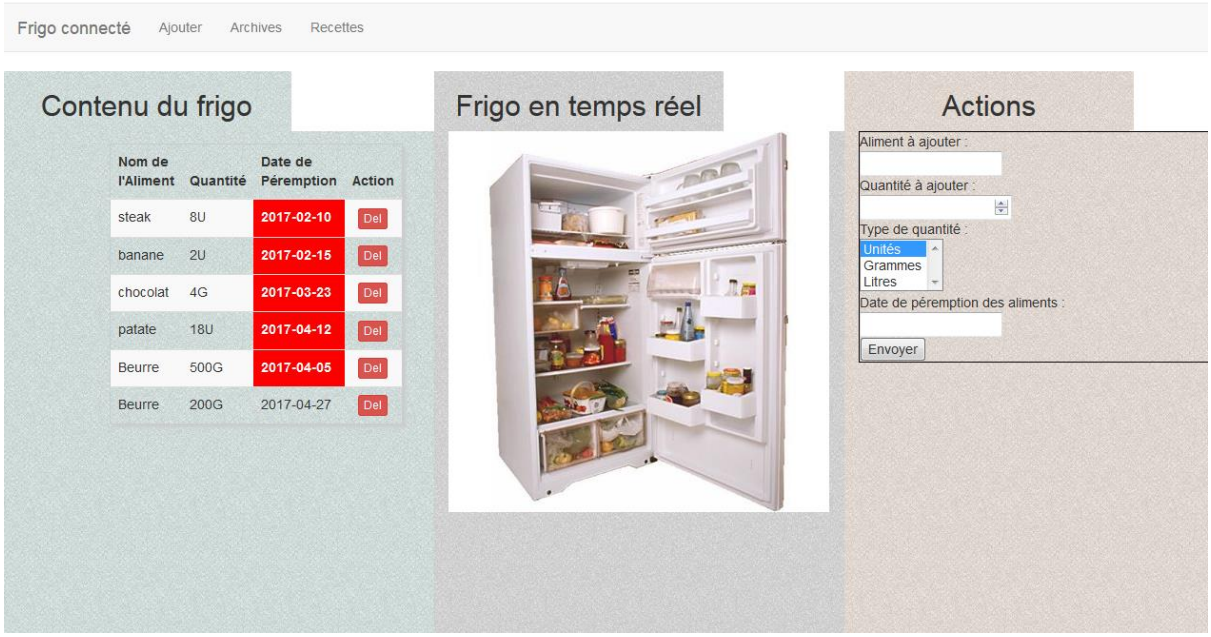
Ce site web test est simplement composé d'un bouton qui appelle le script python quand on clique dessus.

Sprint 5				
Début	05/04/2017			
Fin	28/04/2017			
	Fonctionnalités concernées :		*PF estimés :*	*Terminé le :*
P1	Intégration des éléments sur la carte Raspberry Pi 3		13	*28/4*
P1	Design du site web		13	*28/4*
P1	Intégration du script sur l'application web		13	*28/4*
			Total	39
	Grille d'avancement :			
Date de mélée	*5/4*	*12/4*	*19/4*	*28/4*
PF réalisés	0	0	0	39
PF restants	39	39	39	0

Le Sprint 5 a été consacré à une refonte du design de l'application web, ainsi que l'intégration de tous les élements sur la carte Raspberry PI 3.

Il a été décidé de fusionner l'ensemble des fonctionnalités sur une page, afin d'obtenir un gain énorme en ergonomie. Nous avons pour cela utilisé le framework jQuery, afin de permettre la modification dynamique d'éléments au sein de la même page.

B. Version finale



Comme le montre cette capture d'écran, l'application web se décompose en 3 colonnes.

A gauche, le contenu du frigo, récupéré avec le code de l'ancienne page "consult_frigo.php". Les dates s'affichant en rouge indiquent une date de péremption dépassée. Un clic sur le bouton d'action permet de supprimer l'aliment correspondant.

Au centre, l'image du frigo (ici image de substitution) récupérée via le code Python depuis la caméra intégrée. Nous avons décidé de faire prendre à la caméra une photo toutes les 5 minutes, l'image du frigo étant alors actualisée avec un retour utilisateur.

A droite, les différentes fonctionnalités choisies via la barre de navigation supérieure. A l'écran, le formulaire d'ajout d'aliment. Il est aussi possible de consulter les archives et les recettes enregistrées.

Outils utilisés

Le stockage des données :

Les solutions possibles

On pense tout de suite à utiliser une base de données pour stocker le contenu du frigo ainsi que les recettes, mais il existe de nombreux base de données, toutes différentes les unes des autres. Des relationnelles, des orientées objets ou documents, orientées graphes… Nous avons commencé par utiliser ce que nous connaissons déjà, une base de données SQL, puis nous avons commencé à regarder si on pouvait passés à une base de données NoSQL.

Une base de données MySQL

En vue de nos connaissances actuelles, la solution la plus simple pour stocker des données est d'utiliser une base de données SQL. Il nous faut une base de données légère, compatible avec Node.js, gratuite, et qui s'installe sur Raspberry Pi. PostgreSQL fut un très bon candidat.

Une base de données MongoDB

MongoDB, que nous avons découvert pour l'occasion, répond elle aussi à toutes nos attentes. Il est d'autant plus simple de l'utiliser grâce à son contexte NoSQL. On stocke plus des entrées mais des documents, de la forme que l'on souhaite, avec très peu de restrictions !
De plus MongoDB, grâce à son module Mongoose permettant de s'adapter à Node.js est très rapide et très léger. C'est parfait pour le faire tourner sur un Raspberry Pi. On peut imaginer - si le projet continuait et devenait ouvert au grand public - un très grand nombre de client sans faire ralentir la base de données grâce à la scalabilité de MongoDB.

La solution retenue :

Nous avons choisi d'utiliser une base de données MySQL qui fut très facile à installer sur le Raspberry Pi et très simple d'utilisation. Malgré les avantages proposées par MongoDB,

nous n’avons pas pu l’exploiter parce que le temps attribué au projet nous ne permettais pas de se familiariser avec ce nouveau concept de base de données.

Structure du site et accès aux données stockées :

Les solutions possibles

Un site en PHP

Aux regards des connaissances déjà acquises, le plus simple serait de développer un site en PHP. Pour rendre un site développé en PHP accessible depuis le réseau sur notre Raspberry Pi, nous avons besoin d’un serveur HTTP, Apache par exemple, de PHP, et d’une base de données, Mysql ou autre SGBD. Installer tout ça sur le petit Raspberry Pi rendrait très lourd l’application.

Un site en Node.js

Pour rendre accessible un site développé en Node.js, nous avons juste à installer Node.js sur le Raspberry Pi, qui est très léger comme application. De plus, nous n’avons pas besoin d’installer de serveur HTTP comme “Apache” puisque l’application permet de se mettre en écoute sur un port toute seule.

Malheureusement, Node.js est une technologie que nous ne connaissons pas encore, qui est en plein essor sur le marché du développement WEB. Il serait très intéressant d’avoir une expérience avec cette nouvelle technologie qui est une dérivée du JavaScript utilisable comme serveur WEB.

Un site avec jQuery

jQuery est une bibliothèque javaScript libre et utilisable sur de nombreuses plateformes, créée pour faciliter l'écriture de scripts dans le code HTML des pages web. Cette bibliothèque inclut de nombreuses fonctionnalités comme Ajax, la manipulation facile des CSS, et de nombreux plugins.

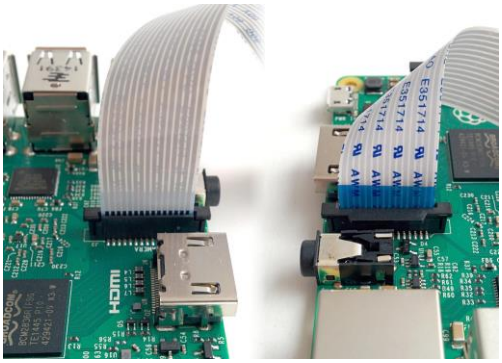
La solution retenue : Un hybride PHP/jQuery

jQuery est un outil très puissant, mais comme beaucoup d’outils, nous ne l’avions pas étudié lors du cursus. Devant le peu de temps à notre disposition pour appréhender les concepts introduits par cette bibliothèque, nous n’avons pu qu’effleurer les possibilités.

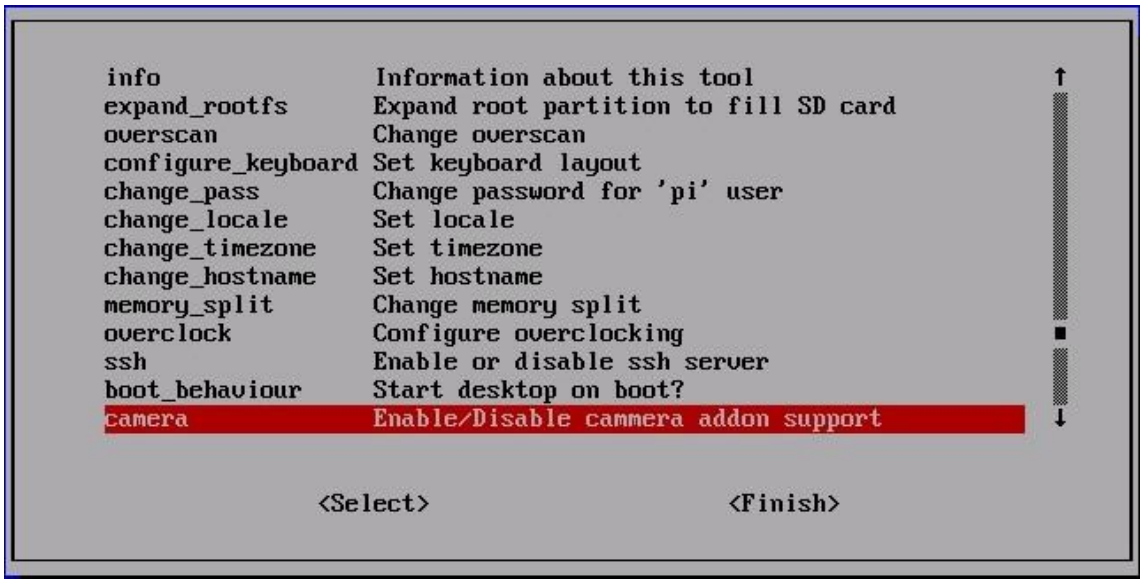
Concernant la structure de l’application web, la page d’accueil est une page HTML dans laquelle est généré du code PHP, localisé dans certaines balises , via la fonction Load de jQuery.
Côté design, nous avons opté pour le framework CSS Bootstrap, qui amène avec lui de très nombreuses améliorations graphiques. Cela nous a permis de réaliser facilement la barre de navigation, ainsi que le découpage en colonne du contenu.

L’acquisition des photos avec la Picamera :

Dans cette partie, nous avions le choix entre plusieurs types de caméras conçues spécialement pour la raspberry pi 3. Notre choix s’est tourné vers la Picamera vu qu’elle permet le rajout d’un module complémentaire pour la vision nocturne. Nous avons décidé de présenter cette partie sous forme de tutoriel, pour vous aider dans la configuration de cette dernière ainsi que dans la correction des bug que vous pouvez rencontrer.



L’installation de la Picamera est très simple : il suffit de brancher la nappe dans le port dédié et d’activer le module sur la raspberry grâce à la commande “sudo raspi-config”.



Ensuite, il faut que vous exécutiez les commandes suivantes afin d’installer les bibliothèques (“library”) :

- sudo apt-get update
- sudo apt-get install python-picamera
- sudo apt-get install python3-picamera

Une fois les commandes exécutées, nous allons devoir créer un script python qui va nous permettre de tester notre picamera et réaliser des captures. L'API python de la picamera est disponible sur internet et est très simple à exploiter. Le lien suivant présente l'intégralité de la documentation du python-picamera : picamera.readthedocs.org.

Ci-dessous un exemple de script python qui permet de tester la picamera après sa configuration :

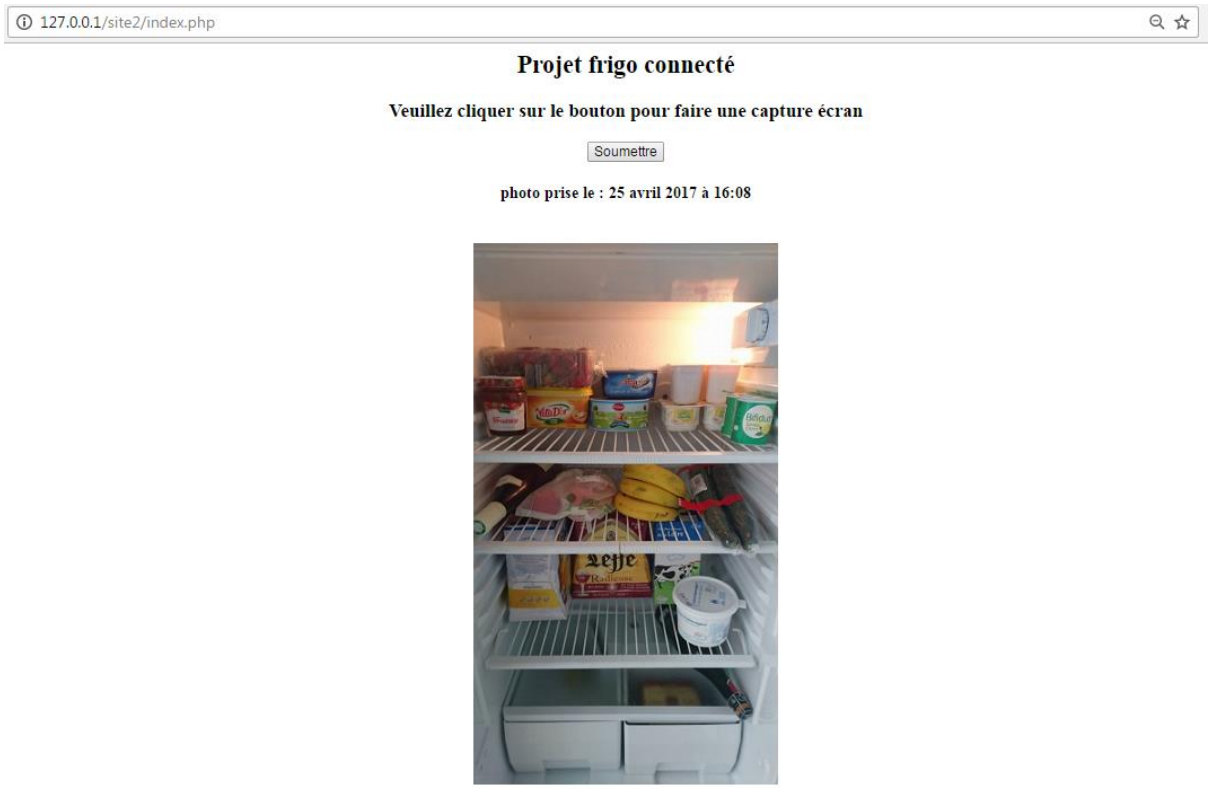
```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.resolution = (640, 480)#définir la résolution souhaiter
camera.start_preview() #Demarre la lecture video
sleep(3) #sleep(X) : Camera allumee pendant X secondes
for i in range(5):#boucle pour repeter 5 fois
    sleep(3)#attente de 3 secondes entre chaque prise de photo
    camera.capture('/home/pi/Documents/Camera/image%s.jpg' % i)#photo capturer et enregistrer
camera.stop_preview() #Coupe la camera
exit()
```

Si le script s'exécute correctement vous êtes censé voir un affichage en temps réel qui dure quelques secondes, puis 5 photos prises et enregistrées dans le répertoire indiqué.

Maintenant que notre programme marche et notre caméra fonctionne bien, nous allons créer une simple application web qui aura comme rôle d'appeler notre script à partir d'un navigateur web. La tâche semble simple mais son développement sous raspbian et le serveur apache la rend un peu compliqué. Sous unix nous ne sommes jamais à l'abri des erreurs liées aux droit de lecture et d'écriture ainsi que quelques problèmes de compatibilités.

Dans cette étape, j'ai créé un petit site en php avec un bouton qui appelle le script python quand on clique dessus.

Si vous cliquez sur le bouton "Soumettre" le code php fait appel au script python, qui, de son toure démarre la Picamera et réalise une capture. Cette photo est enregistrée dans le même répertoire que notre programme et appelée ensuite pour être affichés sur notre application. l'image ci dessous montre le résultat obtenue



Ce travail servira ensuite à intégrer le programme python dans notre application web. Ce dernier va nous permettre d'accéder au contenu de notre frigo à partir d'un navigateur web à n'importe quel moment et quelque soit notre position.

Améliorations possibles

- ***Une implantation massive de jQuery avec un minimum de PHP***

L'application en l'état souffre de nombreux défauts perfectibles. La validation d'un formulaire, même ajouté dynamiquement, entraîne une redirection vers une page PHP dédiée à transmettre la requête SQL, qui elle-même retourne à la page d'accueil après ce traitement.

Avec une meilleure maîtrise de jQuery, il nous aurait été possible de faire des appels avec envoi de données à ces pages PHP, mais sans cette redirection qui n'est pas optimale.

- ***Une refonte des formulaires d'actions***

Comme indiqué plus bas, au début du projet l'application Web était constituée d'une multitude de pages PHP, chacune servant à gérer une fonctionnalité du frigo. Certains ont été intégrées avec succès dans la page d'accueil après le passage en jQuery, comme la page de suppression d'aliment du frigo, transformée en bouton "Supprimer".

Dans ce même esprit, nous aurions pu imprimer une liste de courses à partir des recettes existantes d'un simple clic à partir de l'affichage table, au lieu de la solution actuelle qui est le remplissage d'un formulaire.

Autre amélioration concernant la liste de courses, l'intégration d'une zone de texte en "drag and drop" acceptant recettes existantes, aliments archivés, et nouveautés à remplir, pour un export rapide et efficace.

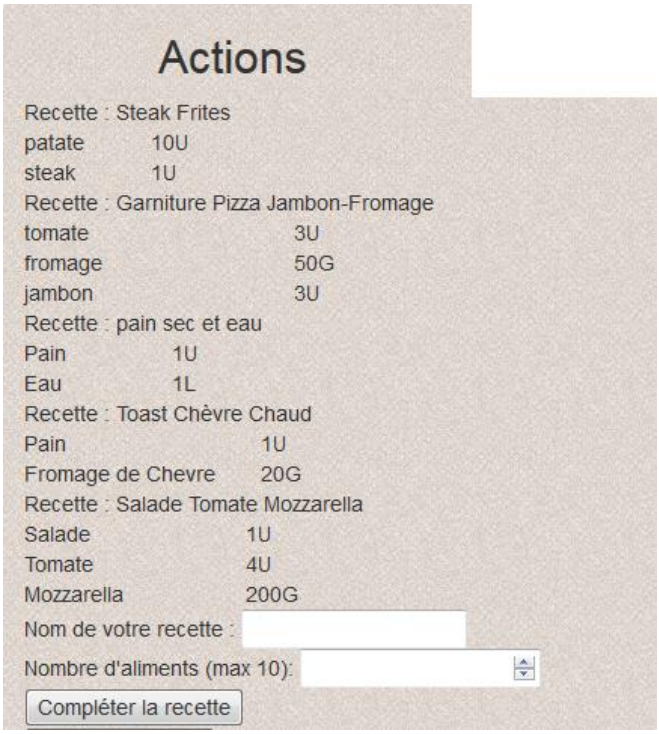
- ***Une portée plus grande pour l'application web***

Actuellement, il n'est possible de se connecter sur la carte embarquée que depuis le réseau local sur lequel elle est installée, qu'il soit Ethernet ou Wifi. Une amélioration possible serait de faciliter l'accès à distance à l'application web. Un utilisateur connecté en 3G sur son téléphone mobile pourrait ainsi obtenir une liste de courses directement dans un magasin.

Problèmes rencontrés

- ***Une mauvaise cohabitation entre jQuery et le CSS***

L'ajout dynamique d'éléments avec jQuery conduit dans certaines conditions (version de navigateur, principalement) à la non application des codes CSS, donnant des résultats peu flatteurs comme ceci :



- ***Droits d'accès aux fichiers***

Plusieurs restrictions liées aux droits d'accès attribués par debian ont empêché l'exécution des scripts python par l'application et le serveur Apache. Il a fallu changer les droits des répertoires utilisés par le serveur web.

- ***Problème matériel***

La Picamera a cessé de fonctionner en pleine phase de tests (composant défectueux).

Le Frigo connecté

Résumé du projet :

Réalisation d'un "frigo connecté", application web placée sur une carte embarquée Raspberry Pi 3 et accessible depuis n'importe quel navigateur internet par l'utilisateur. Cette application, réalisée avec PHP, jQuery et Bootstrap, possède de nombreuses fonctionnalités de contrôle, tel que la visualisation en temps réel du contenu, l'ajout et le retrait d'aliments, la création de liste de courses ou le stockage de recettes de cuisines.

Mots-clés :

Frigo, connecté, Application web, Raspberry Pi, Caméra, PHP, jQuery, Python, Serveur web

Summary :

Creation of a "connected fridge", Web application hosted on a Raspberry Pi 3, embedded card, and designed to be easily accessible for the user.

This application, created with PHP, jQuery and Bootstrap, fulfills many tasks, like visualizing the fridge's content in real time, updating it, creating shopping lists or storing custom recipes.

Keywords :

Fridge, connected, web app, Raspberry Pi, Camera, PHP, jQuery, Python, web server