

ANNÉE SCOLAIRE:
2022/2023



RAPPORT DE PROJET

Projet informatique | Parc Aquatique



Remerciements

Projet informatique | Parc Aquatique

Dans un premier temps, nous tenons à remercier toute l'équipe pédagogique du Lycée Privé Saint-Joseph, ainsi que les intervenants professionnels responsables du B.T.S Systèmes Numériques (SN), pour avoir assuré la partie théorique de celle-ci.

Nous adressons nos plus sincères remerciements à nos professeurs d'informatique, **Mr Bruno Carly**, **Mr Eric Peenart** et **Mr Xavier Kazmierczak**, pour nous avoir aidé à finaliser notre projet informatique, ainsi que pour tout le travail de préparation qui a été fait en amont, afin que celui-ci soit cohérent et intéressant.

Nous tenons à remercier tout particulièrement les personnes suivantes, pour l'expérience enrichissante et pleine d'intérêt qu'elles nous ont fait vivre durant ces nombreuses semaines de projet informatique au sein du Lycée Privé Saint-Joseph :

Mme Luce Euchin, notre professeure de mathématiques, **Mme Audrey Lavoie**, notre professeure de culture générale et d'expression , **Mr André Couvelard**, notre professeur d'anglais , **Mr Eric Cryston**, notre professeur de physique appliquée.

Nous soulignons notre gratitude en raison de l'aide que nos professeurs nous ont apportée tout au long de ce projet, même parfois en dehors de leurs heures de cours.



Le laboratoire des étudiants en 2^{ème} année de B. T.S. SN (Système Numérique).

PRÉSENTATION DU PROJET

Introduction.....	6
Répartitions des tâches.....	8
Planning prévisionnel.....	9
Choix des équipements.....	10
Schémas de câblage.....	15
Explication du projet.....	17
Base de données.....	18
Diagramme de déploiement.....	20
Diagramme des cas d'utilisations.....	21
Diagramme de séquence.....	22
Diagramme de classe.....	27

PROGRAMME D'ACCÈS À L'ATTRACTION

Identification des visiteurs.....	31
Programme d'accès à l'attraction.....	32
Lecture du bracelet.....	33
Base de données.....	35
Requête.....	38
Fonctionnement des led.....	42
Programme principal.....	43
Conclusion.....	46

SITE INTERNET PARC

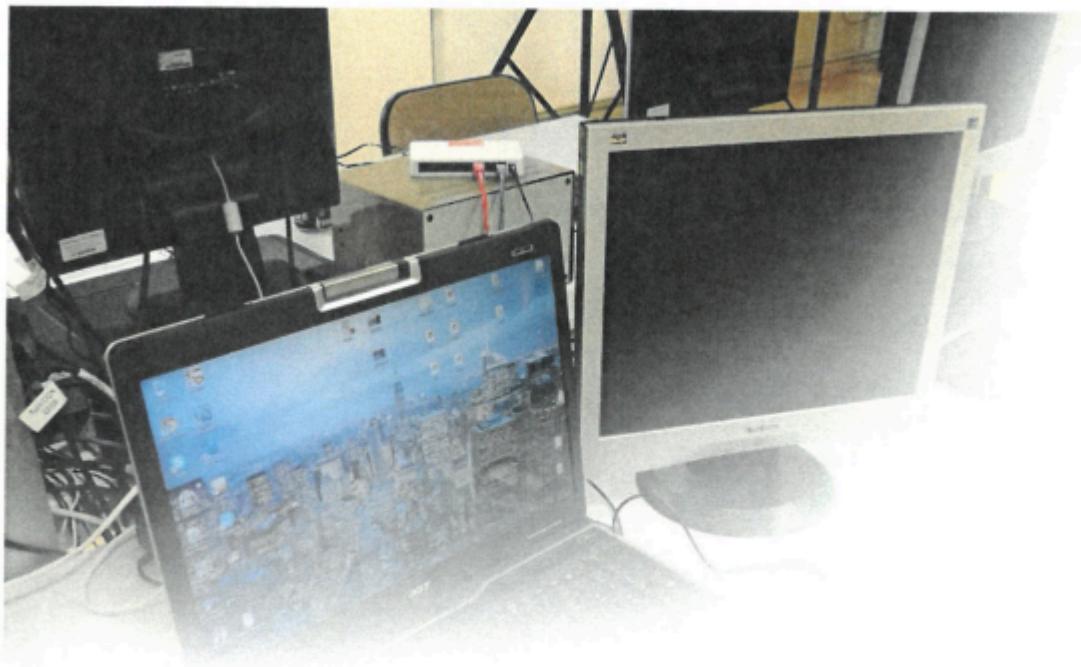
Inscription.php.....	48
Traitemet.php.....	50
Connexion Web.php.....	51
Connexion.php.....	53
Interface.php.....	55
Conclusion.....	56

LOGICIEL ACCÈS AU CASIER

Logiciel d'accès au casier.....	57
C-Lecteur-RFID.....	58
Programme principal.....	60
C-Base de données.....	61
C-Requête.....	63
C-Serrure.....	64

LOGICIEL PRISE DE PHOTO

Logiciel prise de photo.....	67
C-Prise de photo.....	68
Installation GPHOTO2.....	69
Fiche de recette.....	71



PRÉSENTATION DU PROJET

Projet informatique | Gestion RFID parc aquatique

INTRODUCTION

Projet informatique | Présentation du projet

Notre projet est en partenariat avec le parc aquatique O'Gliss Park. Chaque jour au sein du parc, les milliers de visiteurs se verront affecter un bracelet RFID , afin de faciliter leur expérience au sein du parc (exemple: ouverture/fermeture, du casier ; gestion des accès à certaines activités).

Chaque bracelet dispose d'un identifiant unique qui permet d'identifier les visiteurs. On associé à chaque bracelet les informations personnelles de l'individu (nom, prénom, taille). Par exemple, l'"information "taille" autorise l'accès à l'activité si la condition de taille est validée (certaines activités nécessitent une taille minimum).

De plus, chaque visiteur disposera d'un identifiant et d'un mot de passe pour se connecter à son espace personnel sur le site internet du parc. Cet espace leur permettra de consulter leur historique d'attractions, y compris les attractions auxquelles ils se sont vu refuser l'accès en raison de leur taille, ainsi que les photos prises automatiquement pendant leurs moments forts dans le parc. De plus, ils pourront également accéder aux informations relatives à l'ouverture et à la fermeture de leur casier.

En parallèle, nous avons prévu une interface Administration depuis le site du parc pour le responsable technique du parc, lui permettant de consulter les différents accès aux attractions et de visualiser les photos prises lors des attractions. Cela lui offrira une vue d'ensemble du fonctionnement du parc, lui permettant ainsi de mieux gérer les opérations.

Lorsqu'un badge sera lu, on interrogera la base de données puis optera pour l'une des solutions suivantes :

Accès à l'attraction

- Le badge est inconnu, et dans ce cas, on l'affiche à l'aide d'une LED rouge.
- Le badge est reconnu, mais l'accès à une attraction n'est pas possible, car la taille est insuffisante. On affiche l'état avec une LED rouge, puis on envoie l'information à la base de données du refus (avec le nom de l'attraction, la date et l'heure).
- Le badge est reconnu et la taille est suffisante. On indique l'accès à l'attraction avec une LED verte.

Accès au casier

- Le badge est inconnu, et dans ce cas rien ne se passe.
- Le badge est reconnu et demande à l'utilisateur d'ouvrir ou de fermer son casier.

Un aspect important de notre projet est sa capacité d'extension à d'autres parcs aquatiques. En développant une solution flexible et évolutive, nous permettrons aux autres parcs de bénéficier également de ce système RFID, améliorant ainsi l'expérience des visiteurs dans l'ensemble de l'industrie.

Ce rapport détaillera le processus de développement de notre projet, les choix technologiques effectués, ainsi que les résultats obtenus lors de sa mise en œuvre.

RÉPARTITIONS DES TÂCHES

Projet informatique | Présentation du projet

Tâches communes au projet :

- S'approprier la modélisation du système.
- Finaliser le modèle U.M.L de l'application.
- Mise en place du projet.
- Gérer la planification.
- Rédiger les documents relatifs au projet.

Tâche Destinés à Poiret Cédric (étudiant n°1) :

- Réalisation du code permettant la lecture du badge RFID afin d'identifier les clients..
- Programme permettant l'accès à l'attraction.
- Synchronisation des informations avec la BDD. (MySql).

Tâche Destinés à Laforge Yann (étudiant n°2) :

Site Web :

- Authentification des utilisateurs
- Consultation des photos
- Consultations des ouvertures/fermetures du casier
- Consultation des activités refusées

Environnement de développement UwAmp :

- Base de données pour stocker les éléments nécessaires et réaliser des requêtes SQL

Tâche Destinés à Vasseur Pierre (étudiant n°3)

- Codage de la serrure permettant l'accès au casier avec lecteur RFID en traitant les informations avec la BDD.
- Codage permettant la prise de photo grâce au détecteur de présence en envoyant à la BDD la date et heure avec l'ID de l'utilisateur.
- Synchronisation des informations avec la BDD.

PLANNING PREVISIONNEL

Projet informatique | Présentation du projet

En commun :

		2	projet Parc Aquatique	62 jours	Lun (Mer 22/01/23	Jeu 23/01/23	Ven 24/01/23	Samedi 25/01/23	Dimanche 26/01/23		orge;Poiret;Vasseur
✓	➡	2.1	Analyse du Sujet	3 jours	Lun 09/01/23	Lun 23/04/23						Laforge;Poiret;Vasseur 1
✓	➡	2.2	Creation du Gantt	3 jours	Jeu 12/01/23	Lun 16/01/23						Laforge;Poiret 3
✓	➡	2.3	Remplir le cahier des charge	2 jours	Jeu 12/01/23	Ven 13/01/23						Laforge;Poiret;Vasseur 3
✓	➡	2.4	Diaporama du Sujet	2 jours	Mer 18/01/23	Jeu 19/01/23						Laforge;Poiret;Vasseur 5
✓	➡	2.5	Revue 1 20H	0 jour	Ven 20/01/23	Ven 20/01/23						3;4;5;6
✓	➡	2.6	Creation diagramme de deploiement	3 jours	Ven 20/01/23	Mar 24/01/23						Laforge;Poiret;Vasseur 7
✓	➡	2.7	Recherche du materiel	6 jours	Mer 25/01/23	Mer 01/02/23						Laforge;Poiret;Vasseur 8
✓	➡	2.8	Creation de diagramme de cas d'utilisation	9 jours	Jeu 26/01/23	Mar 07/02/23						Laforge;Poiret;Vasseur 7
✓	➡	2.9	Revue 2 50h	0 jour	Mar 07/02/23	Mar 07/02/23						8;9;10
✓	➡	2.10	Creation du diagramme de sequence	5 jours	Mer 08/02/23	Mar 14/02/23						Laforge;Poiret;Vasseur 11
✓	➡	2.11	Creation de diagramme de classe	19 jours	Mer 15/02/23	Lun 13/03/23						Laforge;Poiret;Vasseur 12
CALENDAR	➡	2.12	Revue 3 100h	1 jour	Mar 14/03/23	Mar 14/03/23						12;13

Technicien 1 :

2.15	Création code RFID	18 jours	Mer 15/03/23	Lun 02/03/23
2.15.1	Création code identification	17 jours	Mer 03/03/23	Lun 14/04/23
2.15.2	Création programme led	39 jours	Mer 15/04/23	Lun 23/04/23

Technicien 2 :

2.13	Création du site Web	Mer 15/03/2023	Lun 15/05/2023
2.13.1	Création de la base de donnée	Mer 15/03/2023	Lun 10/04/2023
2.13.2	Création des pages Web en HTML	Lun 10/04/2023	Lun 15/05/2023
2.13.3	Création des php pour liaison	Lun 10/04/2023	Lun 15/05/2023

Technicien 3 :

2.14.1	Programme heure casier C++	Ven 24/02/23	Mar 14/03/23
2.14.2	Programme Photo C++	Mer 15/03/23	Mar 30/05/23

CHOIX DES ÉQUIPEMENTS

Projet informatique | Présentation du projet

Afin de pouvoir réaliser ce projet, un certains nombre **d'équipement**, nous ont été confiés :

Equipement technicien 1 :

Lecteur RFID :

Tableau comparatif et choix du matériel :

Caractéristique	RC522	PN532	RDM6300
Fréquence	13,56 MHz	13,56 MHz	125 kHz
Protocoles pris en charge	ISO/IEC 14443A	ISO/IEC 14443A, 14443B, etc.	EM4100, EM4102
Interface de communication	SPI	UART, I2C, SPI	UART, Wiegand
Alimentation	3,3 V	3,3 V	5 V
Prix	9,90 €	12,99 €	16,38 €

AMAO RFID

Compatible avec les fréquences 13.56 MHz.
lecteur RFID sans contact, plug and play.



Bracelet RFID :

Caractéristique	Bracelet MIFARE® Classic 1K	Carte MIFARE® Ultralight	Bracelet EM4100
Fréquence	13,56 MHz	13,56 MHz	RFID 125KHZ
Capacité de stockage	1K (1024 octets)	Moins de 1K (64 à 512 octets)	0.5K (512 octets)
Protocoles pris en charge	ISO/IEC 14443A	ISO/IEC 14443A	EM4100
Sécurité	Niveau de sécurité élevé	Niveau de sécurité bas	Aucune sécurité intégrée
Utilisations courantes	Contrôle d'accès	Événements, contrôle d'accès	Paiement, billetterie
Résistance à l'eau	Oui	Non	Non
Prix	16.90 €	22.60€	23.90€

CHOIX DES ÉQUIPEMENTS (suite)

Projet informatique | Présentation du projet

MicroContrôleur

Type	64Bit	64Bit	64Bit	64Bit
Marque	Raspberry Pi	Banana Pi	Orange Pi	ASUS
Modèle	4 modèle B	M2 Berry	4	Tinker Board S
Processeur	Broadcom BCM2711	Allwinner V40	Allwinner H6	Rockchip RK3288
GPU	VideoCore VI	Mali-450MP2	Mali-T720MP2	Mali-T760 MP4
RAM	4Go	1Go	4Go	2Go
Prix	55€	40€	50€	80€

Ajout d'une Carte SD Transcend 32 Go MicroSDHC au Raspberry.

Relais

Module de support de relais à 2 canaux Pololu basic avec relais 5VDC.



Led

Led vert de 1.95V avec l'ajout d'une résistance de 330 ohm.

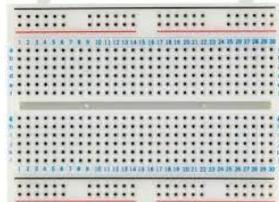


Led rouge de 2.2V avec l'ajout d'une résistance de 330 ohm.



Plaque Lab

Sur cette plaque on peut insérer les éléments électroniques et les fils pour le montage.



CHOIX DES ÉQUIPEMENTS (suite)

Projet informatique | Présentation du projet

Equipement du Technicien 2 :



Développement Base de donnée en SQL sur UWAMP

Equipement du Technicien 3 :

Appareil Photo

Tableau comparatif et choix du matériel :

Marque	CANON	GOPRO	NIKON	PANASONIC
Modèle	EOS 500D	HERO9	COOLPIX S6800	DMC-GX8
Capteur	CMOS APS-C	CMOS	CMOS	MOS
Options	Capture d'Image Trigger Capture	/	Capture d'Image	Capture d'Image
Type	Numérique	Numérique	Compact	Numérique
Compatible avec GPHOTO2	OUI	NON	NON	NON

CHOIX DES ÉQUIPEMENTS (suite)

Projet informatique | Présentation du projet

Capteur étanche à ultrasons SEN0208

Modèle	SEN028	SEN0395
CAPTEUR	ULTRASON	ONDE MILLIMETRIQUE
DISTANCE	25cm à 4,5 mètres	9 mètres
TYPE	Etanche	Pas étanche
PRIX	18,20 Euros	35,90 Euros

MicroContrôleur

Marque	Raspberry Pi	BANANA PI	ORANGE PI	ASUS
Modèle	4 modèles B	M2 BERRY	4	TINKER BOARD S
Processeur	ARM CORTEX-A72	Allwinner V40	Allwinner H6	ROCKSHIP BOARD S
GPU	VideoCore VI	MALI-450 MP2	MALI-T720 MP2	MALI-760 MP4
RAM	8 GO	1 GO	4 GO	2 GO
Type	64 Bits	64 Bits	64 Bits	64 Bits
Prix	95 Euros	40 Euros	50 Euros	80 Euros

Module Relais Simple

Un module de relais 5V est un dispositif composé d'un relais et d'un module de commande. Le relais fonctionne avec une tension de déclenchement de 5V DC et est activé par un signal numérique provenant d'un microcontrôleur ou d'une puce logique.

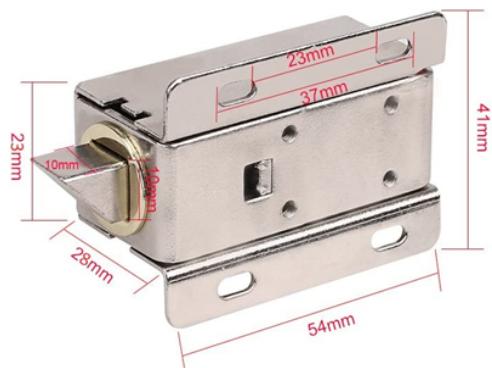


CHOIX DES ÉQUIPEMENTS (suite)

Projet informatique | Présentation du projet

Mini serrure électrique

Une mini serrure électrique 12V est une petite serrure qui utilise une tension électrique de 12 volts pour verrouiller ou déverrouiller des portes, des tiroirs, des boîtes ou d'autres dispositifs.



La serrure est alimentée par une source de courant continu de 12V et peut être contrôlée par des interrupteurs, des boutons-poussoirs, des télécommandes ou des microcontrôleurs. Elle est utilisée dans la domotique, les systèmes de sécurité et les systèmes de contrôle d'accès.

Transformateur :

Le transformateur d'alimentation peut également être équipé de circuits de protection, tels que des fusibles, des disjoncteurs, ou des dispositifs de régulation de tension pour protéger l'appareil électronique contre les surtensions, les surintensités et les fluctuations de tension.

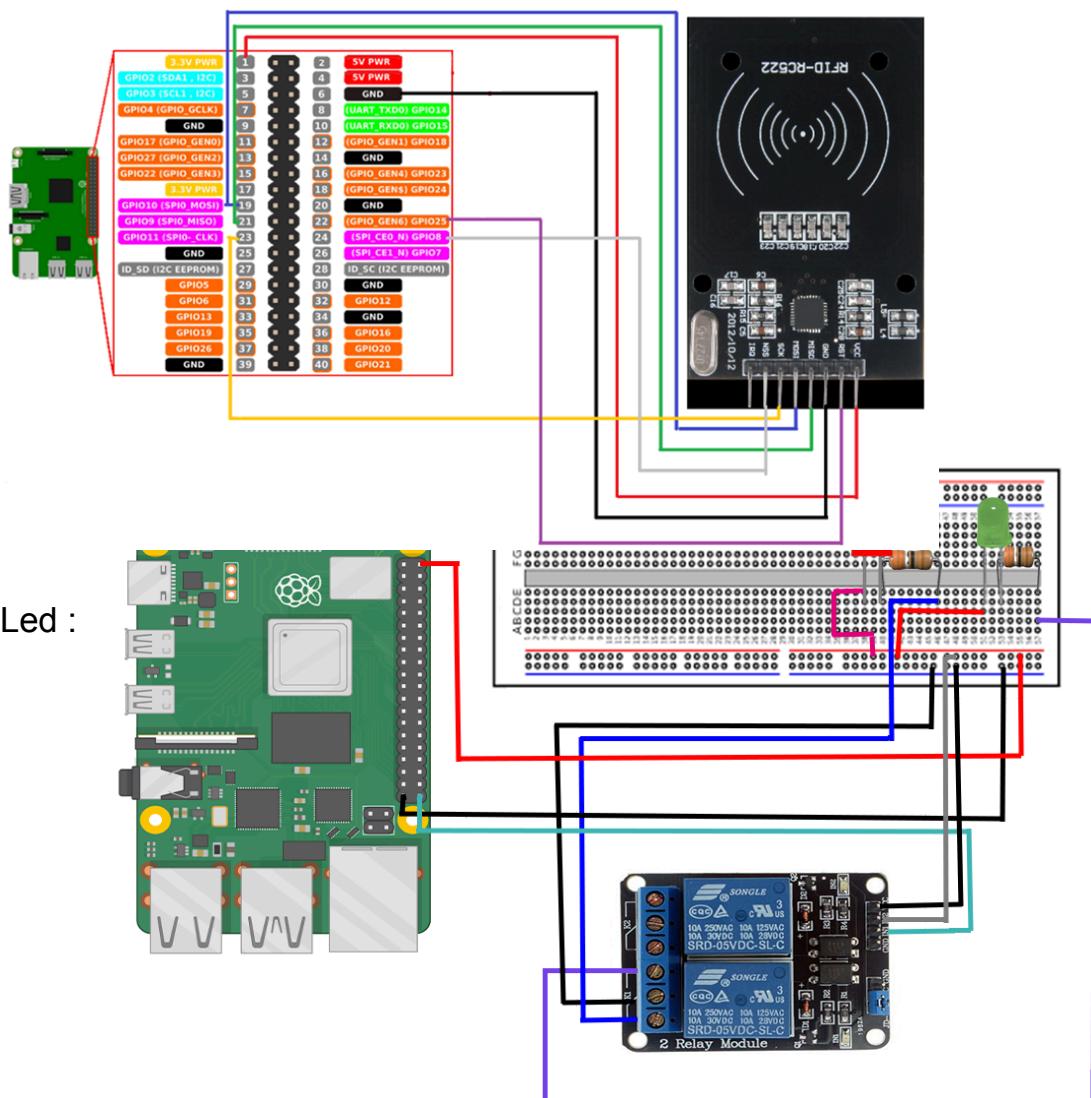


SCHÉMAS DE CABLAGE

Projet informatique | Présentation du projet

Technicien 1

Lecteur RC522 :

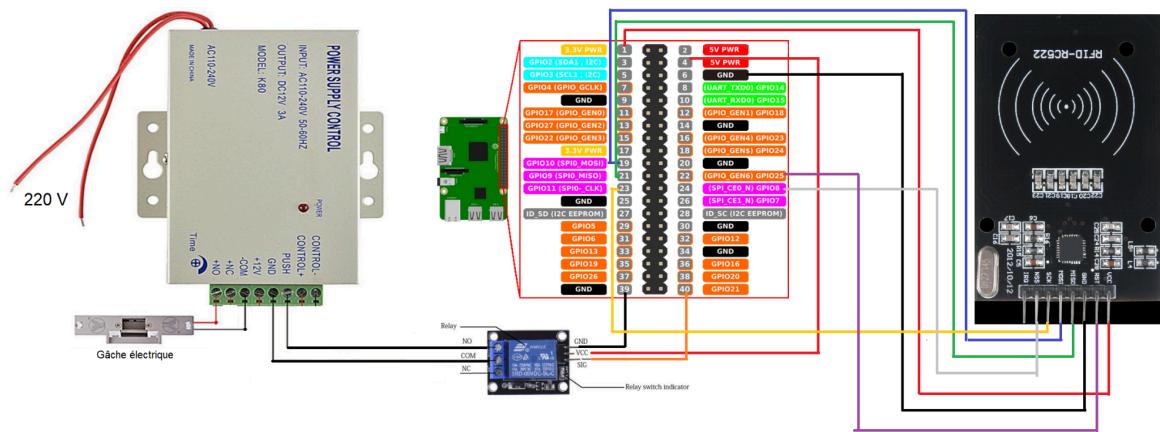


SCHÉMAS DE CABLAGE (*suite*)

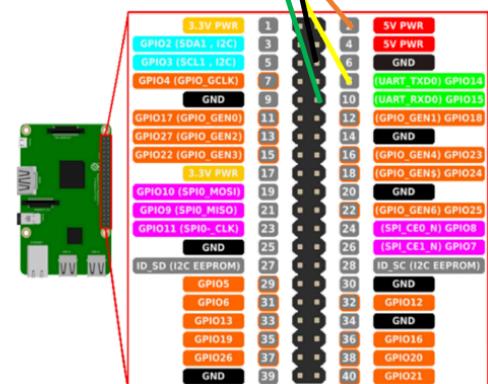
Projet informatique | Présentation du projet

Technicien 3

Système Lecteur RC522 avec serrure



Système Appareil photo avec Module et détecteur de présence.



EXPLICATION DU PROJET

Projet informatique | Présentation du projet

L'objectif de notre projet est donc de pouvoir gérer les accès libres aux attractions et sécuriser ses biens, de pouvoir avoir un accès à ces données depuis le site internet du parc .

Un accès Admin pour les responsables techniques du parc, ce fera depuis le site internet pour consulter les données du parc .

Les bénéfices de ce projet sont nombreux. En effet, grâce à cette gestion RFID , les principaux avantages à tirer sont :

- Une facilité de vérification à l'accès à l'attraction.
- Fonctionnement automatique des casier .
- Gestion automatique des photos.
- Consultation des données .

La base de données centralise toutes les informations relative à notre projet, c'est à dire :

- Information du client (nom,prénom,âge,taille).
- Photo prise durant l'attraction.
- Historique des accès refusés .
- L'état des casiers .
- Historique des visiteurs.

BASE DE DONNÉES

Projet informatique | Présentation du projet

Structure de la base de donnée :

La Base de donnée se compose de 3 tables :



La table visiteurs qui est la table principal contenant les informations du visiteur :

□	1	IdVisiteur	int(11)	Non	Aucune	AUTO_INCREMENT
□	2	nom	varchar(255) latin1_swedish_ci	Oui	NULL	
□	3	prenoms	varchar(255) latin1_swedish_ci	Oui	NULL	
□	4	mot_de_passe	varchar(255) latin1_swedish_ci	Oui	NULL	
□	5	taille	int(11)	Oui	NULL	
□	6	casier	tinyint(1)	Oui	NULL	
□	7	fichier_photo	varchar(255) latin1_swedish_ci	Oui	NULL	
□	8	numero_du_casier	int(255)	Oui	NULL	
□	9	ID	int(11)	Oui	NULL	
□	10	NombreRefus	int(255)	Oui	0	

IdVisiteurs : Est une Clé Primaire qui s'auto incrémente à chaque nouveau visiteur permettra d'avoir un numéro unique pour chaque personne en cas de possible problème avec le badge.

Les informations du visiteurs : Nom, Prénom, Taille, ID du badge, le casier associé , et le nombre d'attractions refusées.

BASE DE DONNÉES (*suite*)

Projet informatique | Présentation du projet

La table Refus

<input type="checkbox"/>	1	ID	int(255)
<input type="checkbox"/>	2	NomAttraction	varchar(255) latin1_swedish_ci
<input type="checkbox"/>	3	HeureRefus	datetime

Permettant de stocker le nom des attractions refusées et à quelle heure elles ont été refusées et de l'associer au numéro du badge.

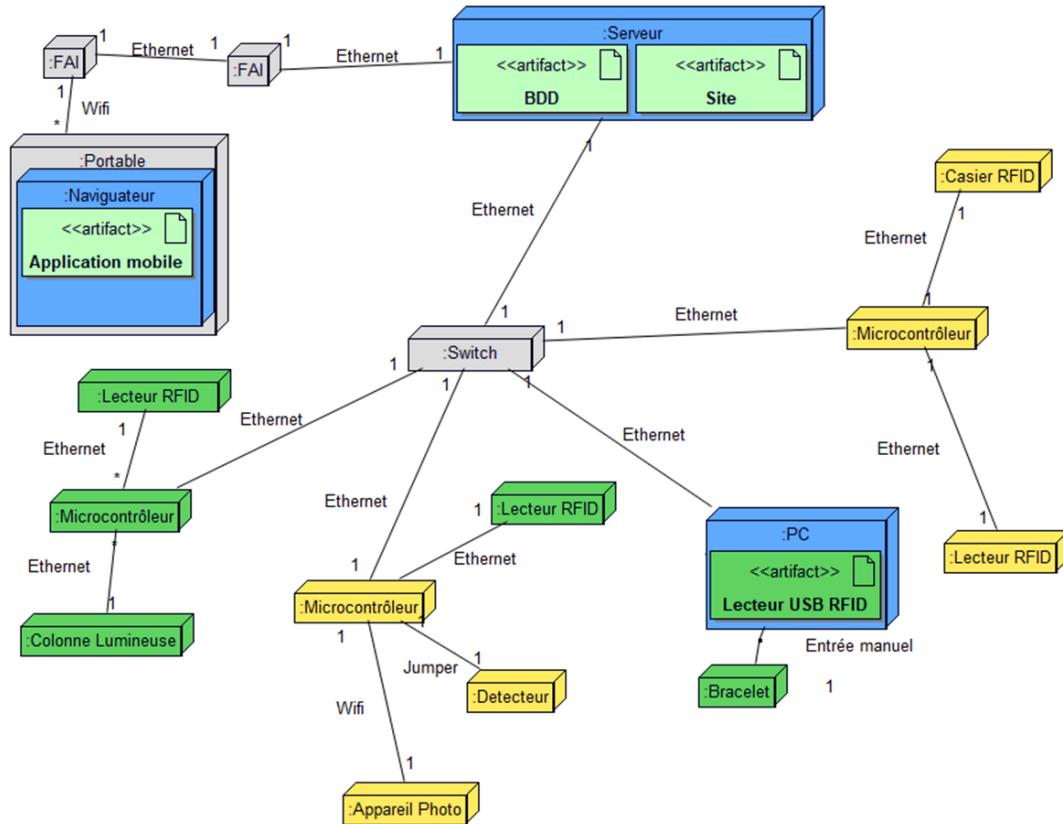
Et une **Table image** permettant d'enregistrer les photo dans la base de donnée sous le type longblob.

Une colonne d'objets volumineux binaires d'une longueur maximale de 4294967295 (2^32 - 1) octets, ou 4 Go de stockage. Chaque valeur LONGLOB est stockée à l'aide d'un préfixe de longueur de quatre octets qui indique le nombre d'octets dans la valeur.)

<input type="checkbox"/>	1	Photo	longblob
--------------------------	---	-------	----------

DIAGRAMME DE DÉPLOIEMENT

Projet informatique | Présentation du projet



Couleur associer au étudiant :

Étudiant n°1 : Vert, Vert pâle.

Étudiant n°2 : Bleu, Vert pâle.

Étudiant n°3 : Jaune.

DIAGRAMME DES CAS D'UTILISATIONS

Projet informatique | Présentation du projet

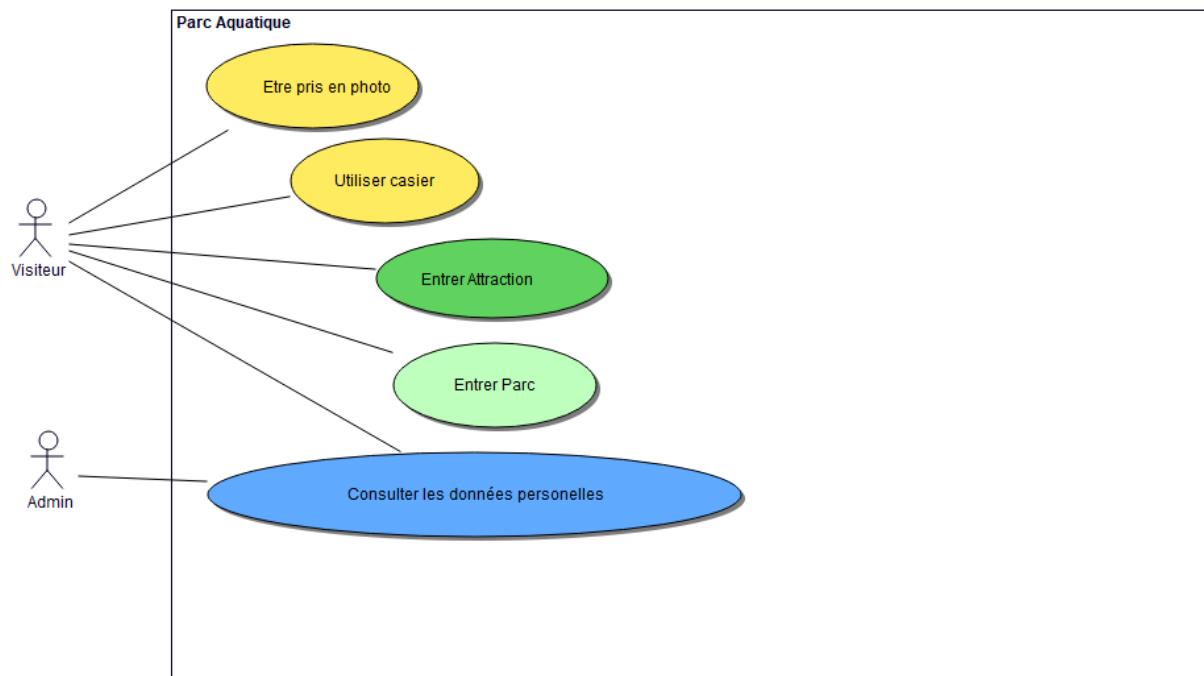


Diagramme des cas d'utilisations.

Couleur associer au étudiant :

étudiant n°1 : Vert,Vert pâle.

étudiant n°2 : Bleu,Vert pâle.

étudiant n°3 : Jaune.

DIAGRAMME DE SÉQUENCE

Projet informatique | Accès à l'attraction

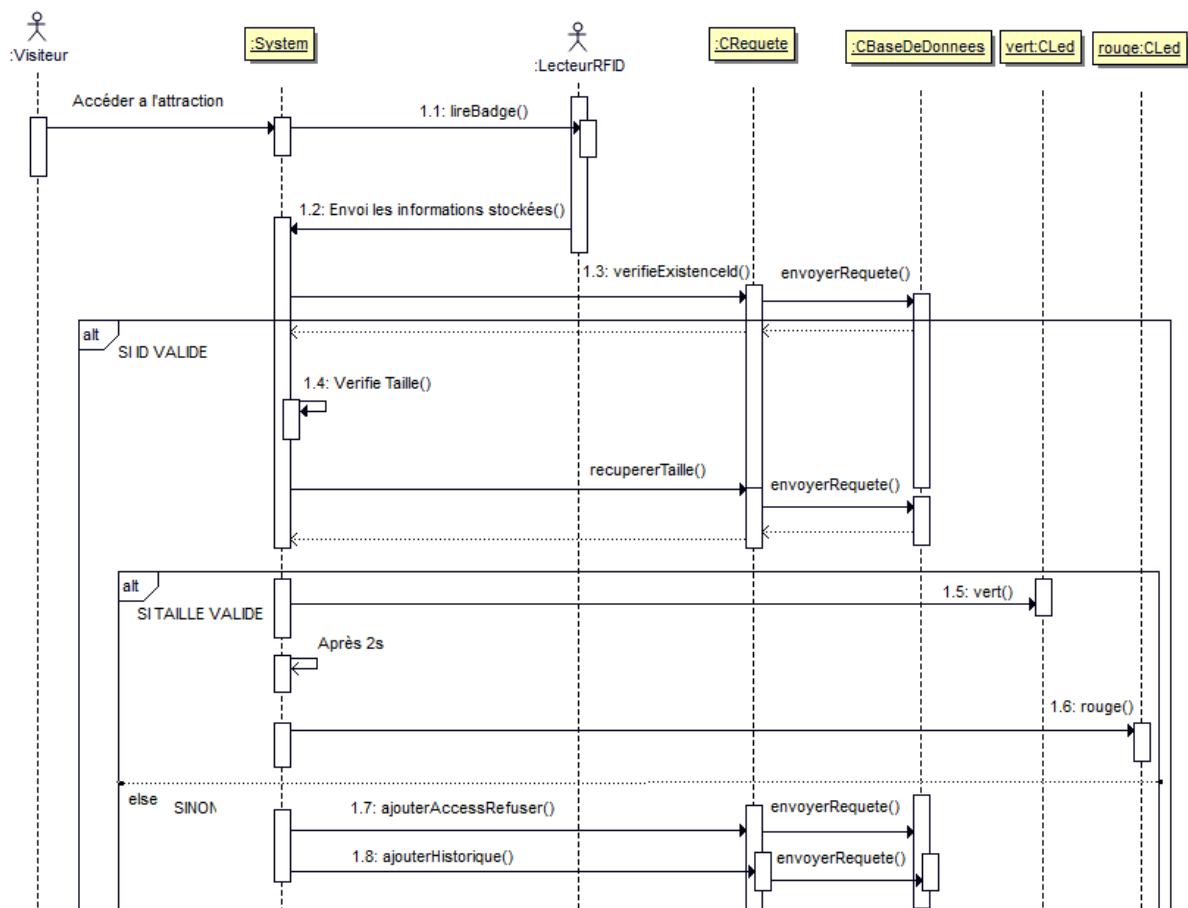


Diagramme de séquence: Accès à l'attraction.

DIAGRAMME DE SÉQUENCE (suite)

Projet informatique | Accès au Casier

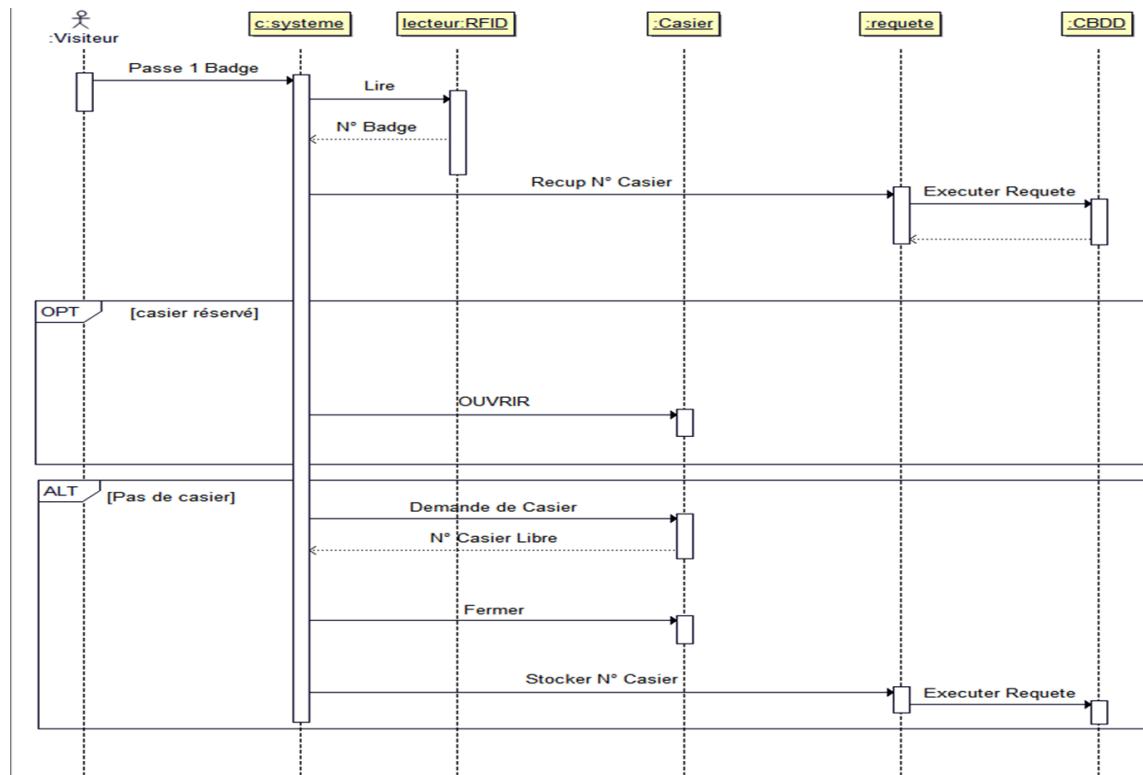


Diagramme de séquence: Accès au Casier

DIAGRAMME DE SÉQUENCE (suite)

Projet informatique | Prise de Photo

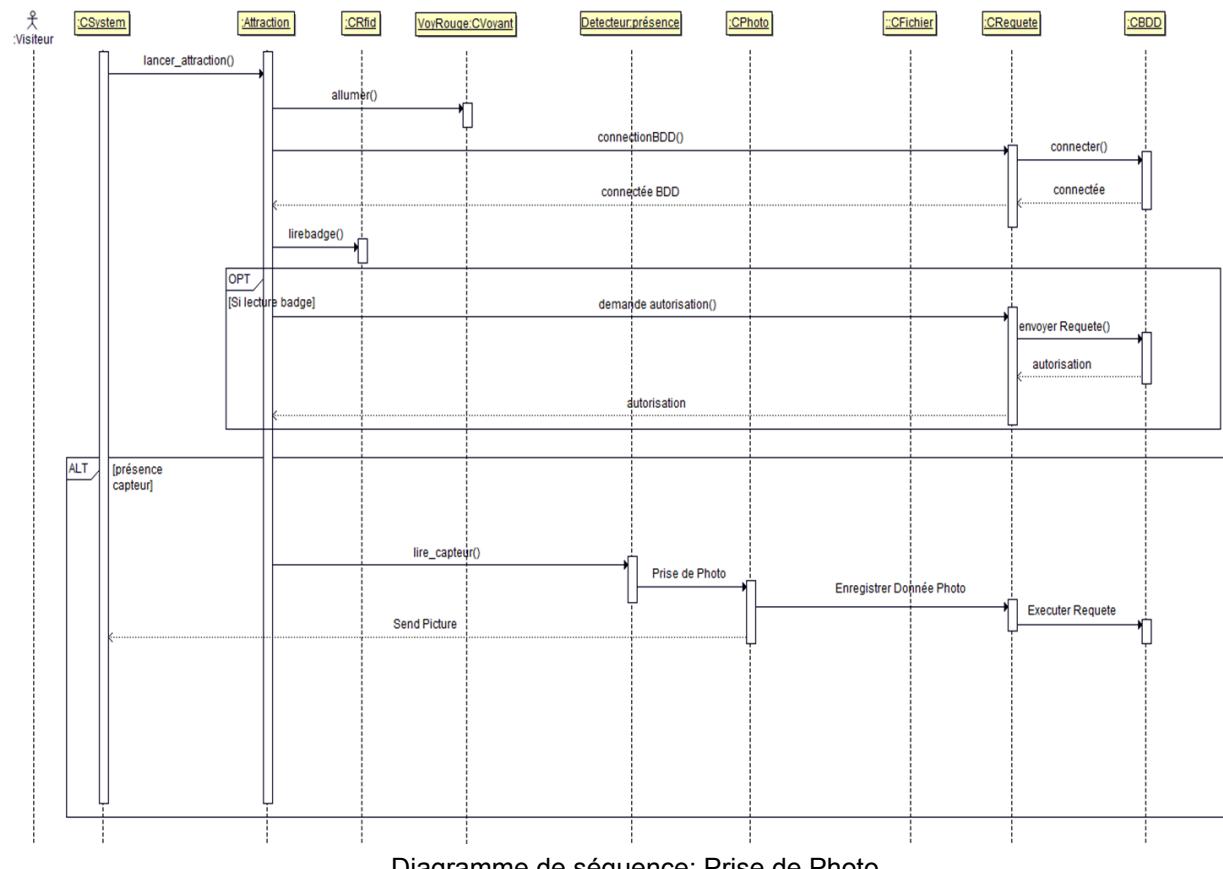


Diagramme de séquence: Prise de Photo

DIAGRAMME DE SÉQUENCE (suite)

Projet informatique | Identification du client

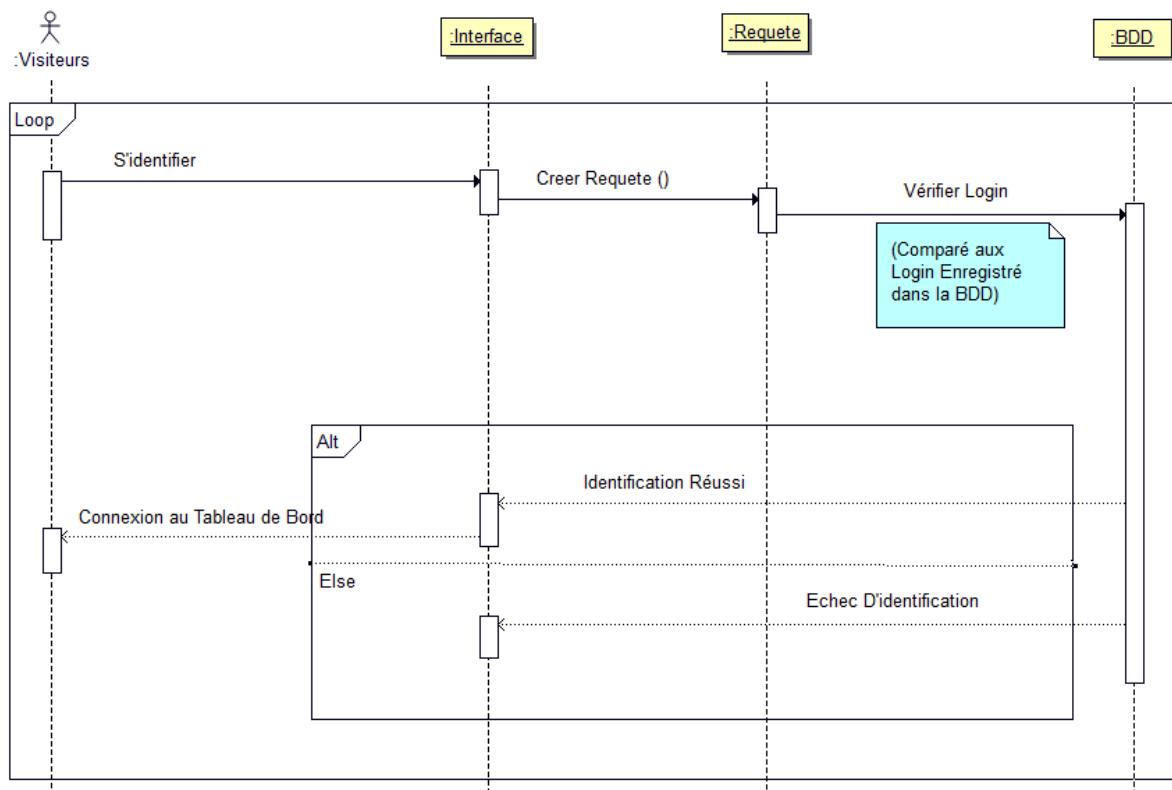


Diagramme de séquence: Identification du client.

DIAGRAMME DE SÉQUENCE (suite)

Projet informatique | Récupération des donnée personnel

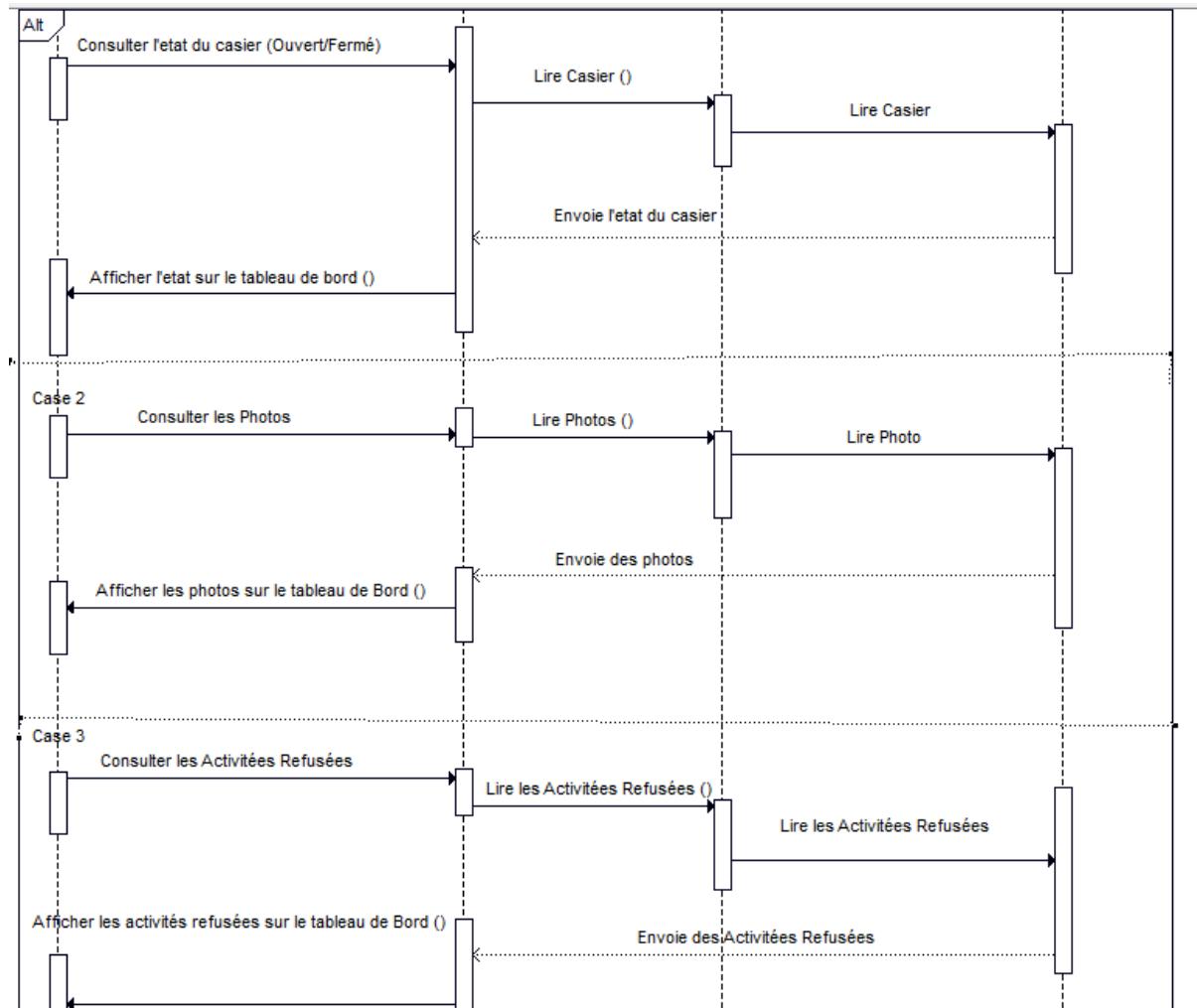


Diagramme de séquence: Récupération des données personnel

DIAGRAMME DE CLASSE

Projet informatique | Accès à l'attraction

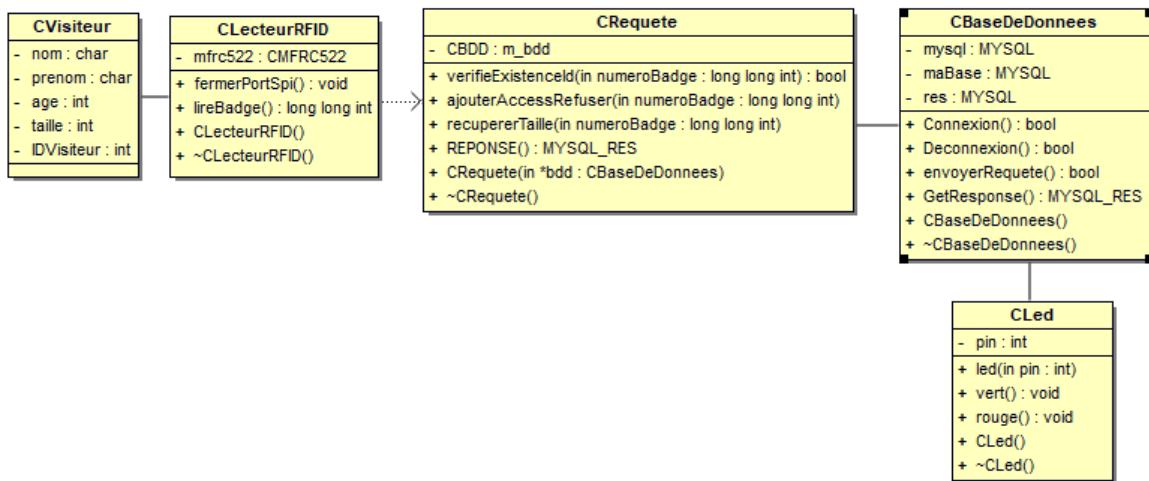


Diagramme de classe: Accès à l'attraction.

DIAGRAMME DE CLASSE (suite)

Projet informatique | Accès au Casier

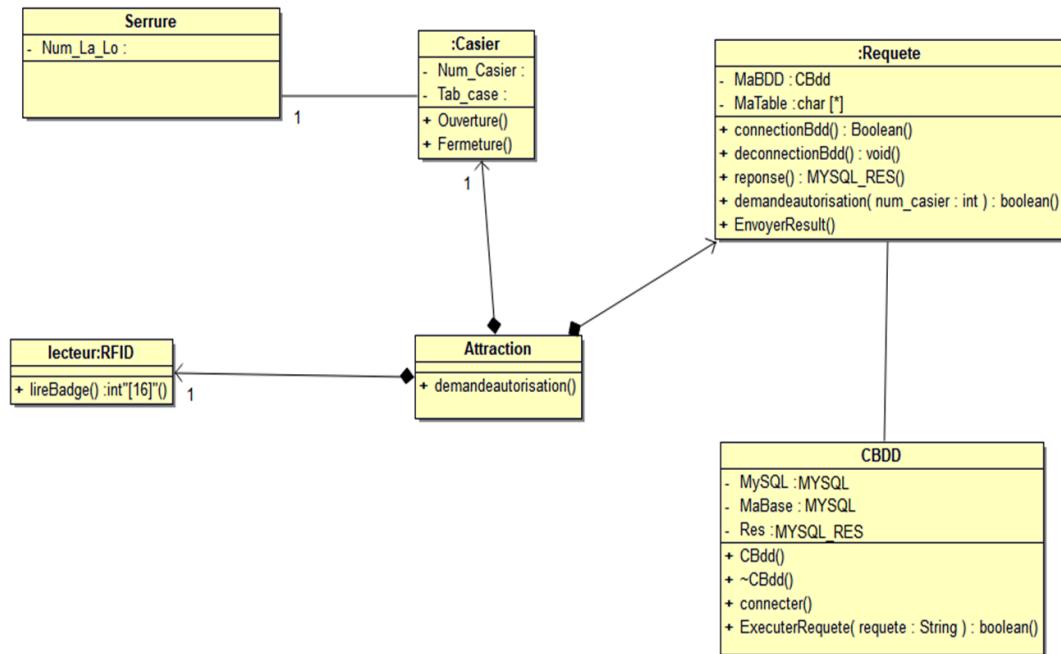


Diagramme de classe: Accès au Casier.

DIAGRAMME DE CLASSE (suite)

Projet informatique | Prise de Photo

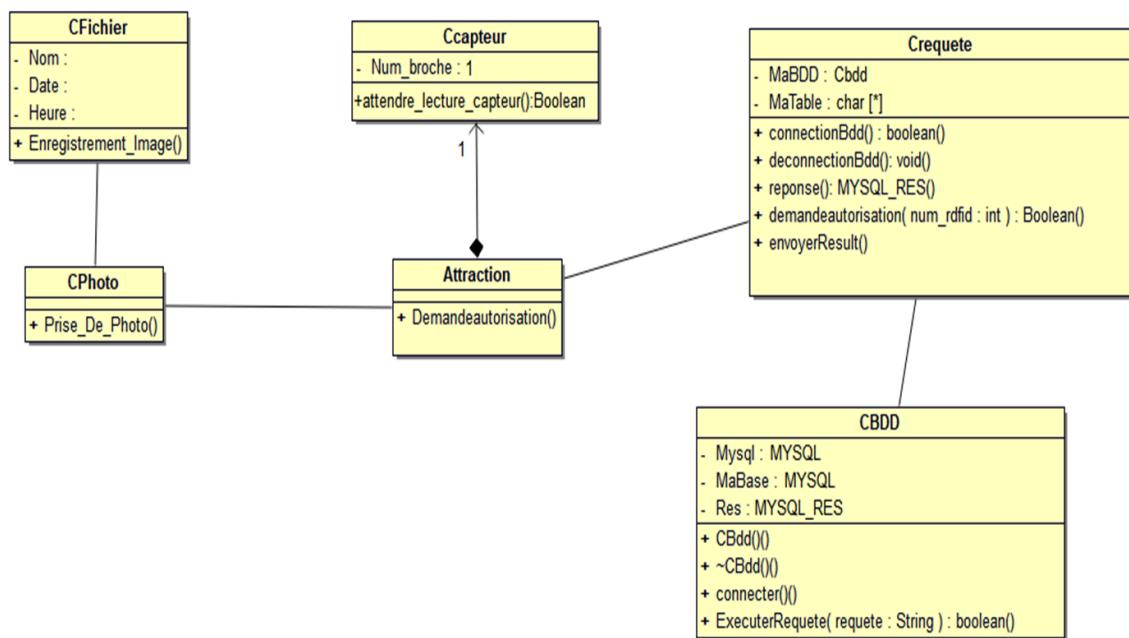
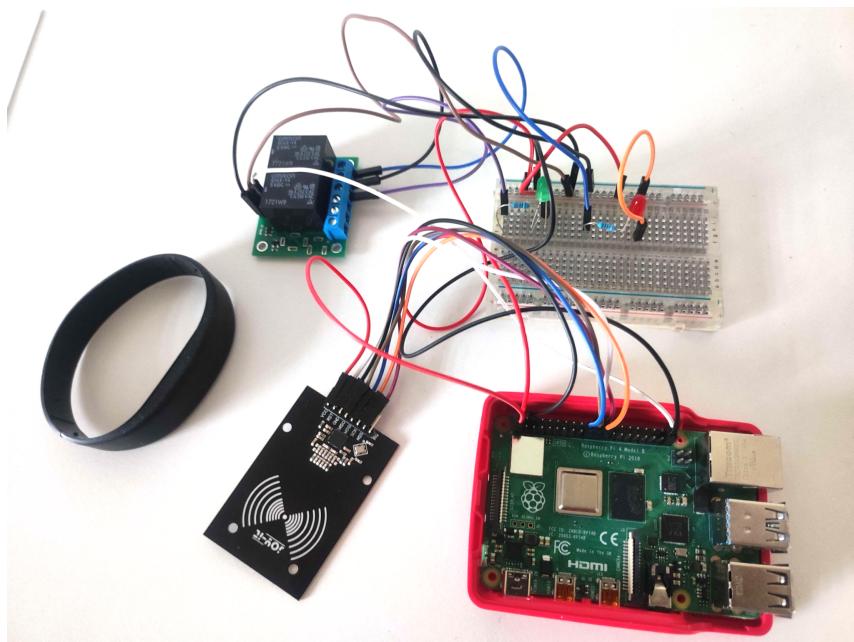


Diagramme de classe: Prise de Photo.

PROGRAMME D'ACCÈS À L'ATTRACTION - Poiret Cédric

Projet informatique | Accès à l'attraction



IDENTIFICATION DES VISITEURS - Poiret Cédric

Projet informatique | Accès à l'attraction

Avant de commencer à créer le programme d'accès à l'attraction, je suis également responsable de la création du code permettant l'identification de chaque visiteur. Ce code est utilisé par plusieurs programmes, notamment dans mon cas pour le programme "**d'accès à l'attraction**", ainsi que les deux programmes du technicien 3 "**accès aux casiers**" et "**prise de photos**".

Le bracelet RFID **MIFARE Classic 1K** contient un identifiant unique par bracelet, donc lors de l'enregistrement du client, l'identifiant du bracelet est associé aux informations du visiteur, ce qui permet d'identifier chaque visiteur au sein du parc.

Le lecteur utilisé pour lire chaque bracelet RFID pour les différents programmes est le **lecteur RC522** pour le contrôler, il est alimenté et connecté aux broches du GPIOs du raspberry(voir schémas de câblage).

La classe **CLecteurRFID.h** permet de retourner l'entier du badge lue et mettre fin à la communication entre le raspberry et le lecteur. Ensuite, une fois le numéro du bracelet récupéré, il faut établir la connexion avec la base de données pour effectuer une **requête** qui va déterminer si le badge est valide ou non.

Dans les différents codes de mon programme "**d'accès à l'attraction**", j'explique le procédé et la manipulation en détaille le traitement pour chaque information pour l'identification du visiteur.

Une fois ce code établi, on peut effectuer la suite du code en rapport avec la création de son programme.

PROGRAMME D'ACCÈS À L'ATTRACTION - Poiret Cédric

Projet informatique | Accès à l'attraction

Présentation

Ce programme permet de contrôler les accès aux différentes attractions en fonction de leur taille.

Fonctionnement

Ce système informatique permet, à partir de sa taille, d'avoir un état sur le passage à une attraction, donc il fonctionne de façon autonome sans avoir besoin d'avoir un employé.

Le logiciel permet donc de :

- D'identifier les visiteurs avec son bracelet.
- D'affecter un état à l'aide d'une led (vert ou rouge) sur son passage.
- D'archiver les refus aux attractions à la base de données (date, nom de l'attraction).

LECTURE DU BRACELET - Poiret Cédric

Projet informatique | Accès à l'attraction

Cette classe a pour but de lire le numéro du bracelet RFID du visiteur et fermer la communication SPI, cela fait partie du premier code nécessaire pour exécuter le reste du programme.

Elle est composée de plusieurs méthodes, une méthode représente une **action** précise.

Pour ce code, nous avons essentiellement deux méthodes, une s'occupe de retourner la valeur décimale du bracelet, et l'une ferme la connexion SPI (**Serial Peripheral Interface**).

Déclaration de la classe CLecteurRFID.h :

```
#ifndef CLECTEURRFID_H
#define CLECTEURRFID_H
#include "CMFRC522.h"
#include <pigpio.h>

class CLecteurRFID
{
public:
    CLecteurRFID();
    ~CLecteurRFID();
    void fermerportspi();
    long long int lirebadge();

private:
    CMFRC522 mfr522;
};

#endif
```

- Le code inclut les en-têtes nécessaires : "**CMFRC522.h**" pour la gestion du lecteur RC522 et "**pigpio.h**" pour la gestion des broches GPIO sur le Raspberry.
- La déclaration `fermerportspi()`, ferme la connexion SPI utilisée par le lecteur.
- La méthode `lirebadge()` retourne un `long long int` représentant le numéro du bracelet en décimale.
- La déclaration de `mfr522` en privé permet de contrôler l'accès à l'objet du lecteur RC522.

LECTURE DU BRACELET (suite) - Poiret Cédric

Projet informatique | Accès à l'attraction

Code source CLecteurRFID.cpp

```
void CLecteurRFID::fermerportspi()
{
    gpioTerminate();
}
```

Le méthode appelle la fonction `gpioTerminate()` de la bibliothèque **pigpio**.

```
long long int CLecteurRFID::lirebadge()
{
    long long int decimalValue=0;
    int rep=1;
    // Initialiation PCD
    mfrc522.PCD_Init();
    while(rep==1) {
        if (mfrc522.isNewCardPresent()==0) { // Détection d'un badge RFID au niveau du lecteur
            // std::cout << "Nouveau badge détecté..." << std::endl;
            delay(5000);
            if(mfrc522.PICC_ReadCardSerial());
            uint8_t nuidPICC[5];
            for (int i = 11; i < 16; ++i)
            {
                nuidPICC[i - 11] = mfrc522.uid.uidByte[i];
            }

            printf("\n");

            for (int i = 0; i < 4; ++i) {
                decimalValue += nuidPICC[i] << (8 * i);
            }
            return decimalValue;
        }
    }
}
```

Le programme se sert de l'objet `mfc522` , la méthode `isNewCardPresent()` vérifie qu'un bracelet soit détecté.

La méthode "PICC_ReadCardSerial()" est utilisée pour lire la trame du bracelet RFID. J'ai constaté qu'à un certain endroit, les octets ne changeaient pas de valeur, ce qui correspondait au numéro du bracelet. À l'aide d'une boucle "for", nous récupérons les octets situés entre la 11e et la 15e position de la trame, qui contiennent la valeur du badge. Ensuite, nous stockons le numéro de série dans la variable "nuidPICC".

Ensuite, pour obtenir la valeur décimale, nous utilisons une boucle "for" qui itère sur les 4 octets du numéro de série. À chaque itération, l'octet correspondant est décalé vers la gauche de $(8 * i)$ bits et ajouté à la variable "decimalValue". De cette manière, "decimalValue" accumule progressivement les valeurs décimales de chaque octet.

BASE DE DONNÉES - Poiret Cédric

Projet informatique | Accès à l'attraction

La classe CBaseDeDonnees joue un rôle crucial dans le programme en facilitant l'interaction avec une base de données. Son objectif principal est d'établir et de gérer la connexion à la base de données, ainsi que d'exécuter des requêtes SQL et de récupérer les résultats.

en prenant l'exemple de la méthode envoyerRequete() est utilisée pour envoyer des requêtes SQL à la base de données. Elle prend en paramètre une chaîne de caractères représentant la requête SQL à exécuter. En cas de succès, elle renvoie **true**, sinon elle renvoie **false**. Cette méthode gère les erreurs et affiche un message d'erreur si la requête échoue.

La classe CBaseDeDonnees

```
#ifndef CBASE_DE_DONNEES_H
#define CBASE_DE_DONNEES_H

#include <mysql/mysql.h>
#include <string>

class CBaseDeDonnees {
private:
    MYSQL mysql;
    MYSQL *maBase;
    MYSQL_RES *res;

public:
    CBaseDeDonnees();
    ~CBaseDeDonnees();
    bool connexion(char *server, char *user, char *password, char *database);
    bool deconnexion();
    bool envoyerRequete(char *requete);
    MYSQL_RES *GetResponse();
};

#endif
```

- L'attribut ***maBase** est un pointeur qui sera utilisé pour pointer vers l'objet MYSQL initialisé lors de la connexion à la base de données.. Il est utilisé pour interagir avec la base de données lors de l'exécution des requêtes.
- L'attribut ***res** lui aussi un pointeur , utilisé pour stocker le résultat des requêtes exécutées sur la base de données. Il permet de récupérer les données résultantes à partir des requêtes.

BASE DE DONNÉES (suite) - Poiret Cédric

Projet informatique | Accès à l'attraction

- La méthode `connexion()` prend en paramètres les informations de connexion (je vais détailler l'identification de l'utilisateur sur la base de donnée dans la partie “**Programme principal.**”).
- `deconnexion()` : la méthode ferme la connexion à la base de données.
- `*GetResponse()` la méthode renvoie le résultat de la dernière requête, Ce résultat peut être utilisé pour extraire les données nécessaires à partir de la base de données

Code source CBaseDeDonnes.cpp

```
bool CBaseDeDonnees::connexion(char *server,char *user,char *password, char *database) {  
    mysql_init(&mysql);  
    maBase=mysql_real_connect(&mysql, server, user, password, database, 0, 0, 0);  
    if (maBase==NULL){  
        return false;  
    }  
    return true;  
}
```

La méthode `connexion()` retourne true si la connexion, c'est bien passé sinon false.

```
bool CBaseDeDonnees::deconnexion() {  
    mysql_close(maBase);  
    return true;  
}
```

On utilise retourne true quand la déconnexion à la base de donnée s'opère.

Projet informatique | Accès à l'attraction

```
bool CBaseDeDonnees::envoyerRequete(char *requete) {  
    int query_state;  
    query_state=mysql_query(&mysql, requete);  
    if (query_state!=0){  
        std::cout << "Erreur lors de l'envoi de la requete : " << mysql_error(&mysql) << std::endl;  
        return false;  
    }  
    res=mysql_store_result(maBase);  
    return true;  
}
```

La méthode `envoyerRequete()` prend en paramètre une chaîne de caractère, la fonction `mysql_query()` s'occupe d'exécuter la requête avec le pointeur sur l'objet “MYSQL” qui représente la connexion à la base de données, et la chaîne de caractère. Puis vérifie s'il y a une erreur lors de l'envoi de la requête ensuite, on récupère le résultat de la requête dans la variable “res”.

```
MYSQL_RES *CBaseDeDonnees::GetResponse()  
{  
    return res;  
}
```

Retourne la valeur de la variable `res`, qui est le pointeur vers l'objet `MYSQL_RES` contenant le résultat de la dernière requête exécutée.

REQUÊTE - Poiret Cédric

Projet informatique | Accès à l'attraction

La classe CRequete joue un rôle essentiel dans le traitement des requêtes vers la base de données. Elle facilite l'interaction avec la base de données en fournissant des méthodes pour effectuer différentes opérations telles que la **vérification de la validité du bracelet**, la récupération de la taille du visiteur.

La classe CRequete

```
-#ifndef REQUETE_H
#define REQUETE_H

#include <string>
#include <vector>
#include "CBaseDeDonnees.h"

class CRequete {
private:
    CBaseDeDonnees* m_bdd;
public:
    CRequete(CBaseDeDonnees* bdd);
    ~CRequete();
    bool verifierExistenceId(long long int numerobadge);
    int recupererTaille(long long int numerobadge);
    MYSQL_RES *REPONSE();
    bool ajouterAccesRefuser(long long int numerobadge);
    bool ajouterHistorique(long long int numerobadge);
};

#endif
```

- L'attribut “**CBaseDeDonnees* m_bdd**” permet à la classe CRequete d'accéder et d'utiliser les fonctionnalités de la classe **CBaseDeDonnees**, en établissant une relation de dépendance entre les deux classes
- Le constructeur **CRequete(CBaseDeDonnees* bdd)** permet d'initialiser un objet CRequete en lui fournissant un pointeur vers un objet **CBaseDeDonnees**, établissant ainsi une relation entre les deux classes pour effectuer des opérations de requête sur la base de données.

REQUÊTE (suite) - Poiret Cédric

Projet informatique | Accès à l'attraction

- La méthode **verifierExistenceId()** permet à partir du numéro de badge qui a été récupérée avec la méthode **lireBadge()** de ClecteurRFID() de vérifier s'il est répertorié dans la base de données.
- La méthode **récupererTaille()** retourne la taille du visiteur.

Taille insuffisante

Si la personne a été refusée par rapport à sa taille si elle est inférieure à celle de l'attraction, on doit ajouter à la base de données l'information du nombre de refus donc c'est cela à quoi sert la méthode ajouter **AccesRefuser()**, et la méthode **ajouterHistorique()** envoi (nom de l'attraction, date heure, id du badge refusé) dans la table **Refus** de la base de données qui va permettre d'avoir un historique.

méthode REPONSE() :

Lorsque la méthode **envoyerRequete()** de la classe **CBaseDeDonnees** est appelée pour exécuter une requête SQL, elle stocke le résultat dans l'attribut **res** de la classe **BaseDeDonnees**. La méthode **REPONSE()** de la classe **CRequete** permet d'accéder à cet attribut **res** et de le retourner. Elle renvoie donc un pointeur vers le résultat de la dernière requête exécutée.

Code source CRequete.cpp :

```
CRequete::CRequete(CBaseDeDonnees* bdd) {
    m_bdd = bdd;
}
```

Permet d'initialiser l'**attribut "m_bdd"** de la classe **CRequete** avec l'**objet BaseDeDonnees** passé en paramètre, établissant ainsi une référence à cet objet pour effectuer des opérations de **requête** sur la base de données.

REQUÊTE (suite) - Poiret Cédric

Projet informatique | Accès à l'attraction

Types de Requêtes

Dans le cadre de ce projet, plusieurs types de requêtes sont utilisés pour interagir avec la base de données. Chaque type de requête représente une action spécifique à effectuer sur la base de données. Voici une description de ces types de requêtes :

Vérification de l'existence d'un ID

Cette requête est utilisée pour vérifier si un identifiant spécifique existe dans la base de données. Elle permet de déterminer si un enregistrement correspondant à cet ID est présent. Cette requête utilise la clause SELECT avec une condition basée sur l'ID recherché.

Récupération de la taille d'un ID

Cette requête est utilisée pour récupérer la taille associée à un identifiant spécifique dans la base de données. Elle permet d'obtenir une information spécifique liée à cet ID. La requête utilise la clause SELECT pour sélectionner la taille correspondante à l'ID recherché.

Ajout d'un accès refusé

Cette requête est utilisée pour mettre à jour le nombre d'accès refusés pour un identifiant spécifique dans la base de données. Elle incrémente le nombre d'accès refusés chaque fois qu'un accès est refusé. La requête utilise la clause UPDATE pour modifier l'enregistrement correspondant à l'ID spécifié.

Ajout d'un enregistrement à d'historique

Cette requête est utilisée pour ajouter un nouvel enregistrement à l'historique dans la base de données. Elle insère les informations pertinentes, telles que l'identifiant, l'heure du refus et le nom de l'attraction, dans la table d'historique. La requête utilise la clause INSERT INTO pour ajouter un nouvel enregistrement.

Chaque requête est composée d'une chaîne de caractères contenant la requête SQL correspondante. Cette chaîne est construite en utilisant des fonctions de manipulation de chaînes pour incorporer les valeurs dynamiques, telles que l'ID ou d'autres paramètres spécifiques à chaque requête.

REQUÊTE (suite) - Poiret Cédric

Projet informatique | Accès à l'attraction

```
bool CRequete::verifierExistenceId(long long int numerobadge) {
    char req[500];
    sprintf(req, "SELECT ID FROM visiteurs WHERE ID='%lld'", numerobadge);
    printf("la requete est %s",req);

    if(m_bdd->envoyerRequete(req)){
        MYSQL_RES* resultat = REPONSE();
        if(resultat != NULL){
            MYSQL_ROW row = mysql_fetch_row(resultat);
            mysql_free_result(resultat);
            if(row != NULL){
                return true;
            }
        }
    }
    return false;
}

int CRequete::recupererTaille(long long int numerobadge) {
    char req[500];
    sprintf(req, "SELECT `taille` FROM `visiteurs` WHERE ID='%lld'", numerobadge);
    printf("la requete est %s",req);

    if (!m_bdd->envoyerRequete(req))
        printf("Erreur");
        return -1; // retourner une valeur d'erreur
    }

    MySQL_RES *resultat = m_bdd->GetResponse();
    MySQL_ROW row;
    int taille = -1; // initialiser la variable à une valeur d'erreur
    while ((row = mysql_fetch_row(resultat))) {
        taille = std::stoi(row[0]); // convertir la chaîne en entier
        // std::cout << "taille : " << taille << std::endl;
    }

    return taille;
}

bool CRequete::ajouterAccesRefuser(long long int numerobadge) {
    char req[500];
    sprintf(req, "UPDATE visiteurs SET NombreRefus = NombreRefus + 1 WHERE ID='%lld'", numerobadge);
    printf("la requete est %s", req);

    return m_bdd->envoyerRequete(req);
}

bool CRequete::ajouterHistorique(long long int numerobadge) {
    char req[500];
    std::string attraction = "Toutatis";
    sprintf(req, "INSERT INTO refus(ID, HeureRefus, NomAttraction) VALUES ('%lld', now(), '%s')", numerobadge, attraction.c_str());
    printf("la requete est %s", req);
    return m_bdd->envoyerRequete(req);
}
```

```
MySQL_RES *CRequete::REPONSE() {
    return m_bdd->GetResponse();
}
```

Pour chaque **requête** pour exécuter la requête définie “return m_bdd->envoyerRequete(req)” permet **d'appeler** la méthode **envoyerRequete()** de **l'objet**

CBaseDeDonnees à travers le pointeur m_bdd dans la classe CRequete et de renvoyer la valeur de retour de cette méthode à l'endroit où la méthode est appelée. Cela permet d'effectuer l'envoi de la requête SQL à la base de données et de gérer le résultat de cette opération dans le contexte de la méthode.

FONCTIONNEMENT DES LED - Poiret Cédric

Projet informatique | Accès à l'attraction

La classe CLed a pour objectif de contrôler deux led. Cette classe offre deux méthodes pour allumer la led verte et rouge. Les méthodes vont servir pour afficher l'état de l'accès à l'attraction en fonction de sa taille.

```
#ifndef CLED_H
#define CLED_H

class CLed {
private:
    int pin_;

public:
    CLed(int pin);
    void Vert();
    void Rouge();
};

#endif
```

Nous avons un attribut privé pin_ qui représente le numéro de la broche GPIO à laquelle la LED est connectée.

Le constructeur CLed() est utilisée pour créer un objet led et spécifier le numéro de la broche GPIO associée à la LED à contrôler.

Puis les deux méthodes pour allumer la led.

Code source associé :

```
#include "CLed.h"
#include <wiringPi.h>

CLed::CLed(int pin) {
    pin_ = pin;
    pinMode(pin_, OUTPUT);
}

void CLed::Vert() {
    digitalWrite(pin_, HIGH);
}

void CLed::Rouge() {
    digitalWrite(pin_, LOW);
}
```

WiringPi est utilisée pour contrôler les broches GPIO.

la méthode CLed() :

On assigne la valeur du pin à l'objet.

On indique au système qu'il souhaite utiliser cette broche.

La méthode "Vert()" allume la LED verte avec la valeur "HIGH", tandis que la méthode "Rouge()" allume la LED rouge avec la valeur "LOW". Dans la configuration avec un relais (voir **schéma de câblage**), la LED rouge est constamment allumée via la borne **NC1**. Lorsque nous activons la LED verte, le relais **NO1** se ferme, permettant ainsi l'allumage de la LED verte.

PROGRAMME PRINCIPAL - Poiret Cédric

Projet informatique | Accès à l'attraction

```
18 int main(int argc, char *argv[]) {
19     CBaseDeDonnees bdd;
20     if (!bdd.connexion("172.29.21.52", "AdminP", "Aparc", "parc_aquatique")) {
21         std::cerr << "Erreur de connexion à la base de données." << std::endl;
22         return 1;
23     }else{
24
25
26         if (gpioInitialise() < 0)
27         {
28             std::cerr << "Port SPI non initialisé." << std::endl;
29         }
30         else
31         {
32             CLecteurRFID monlecteur;
33             int retourhaut =1;
34             while(retourhaut != 0) {
35                 long long int decimalValue = monlecteur.lirebadge();
36                 if (decimalValue != 0)
37                 {
38                     printf("Numero de badge converti en base 10: %lli\n", decimalValue);
39                     CRequete req(&bdd);
40                     bool existe = req.verifierExistenceId(decimalValue);
41
42                     std::cout << "La valeur du booléen est : " << existe << std::endl;
43
44                     if(existe == true)
45                     {
46                         std::cout << "L'ID " << decimalValue << " existe dans la base de données." << std::endl;
47                         int taille = req.recupererTaille(decimalValue);
48                         std::cout << "Taille : " << taille << std::endl;
49
50                         if(taille >= 175)
51                         {
52                             wiringPiSetup();
53                             CLed ma_led(29);
54                             ma_led.Vert();
55                             delay(600);
56                             ma_led.Rouge();
57
58                         }else{
59                             req.ajouterAccesRefuser(decimalValue);
60                             req.ajouterHistorique(decimalValue);
61
62                             break;
63                         }
64
65                     }
66
67                     else
68                     {
69                         std::cerr << "L'ID " << decimalValue << " n'existe pas dans la base de données." << std::endl;
70                     }
71
72                 }
73                 delay(1000);
74                 retourhaut =1;
75             }
76             monlecteur.fermerportspi();
77         }
78     }
79 }
```

Explication du déroulement du programme

Le programme fait fonctionner l'ensemble du projet en regroupant toutes les classes.

PROGRAMME PRINCIPAL (suite) - Poiret Cédric

Projet informatique | Accès à l'attraction

Connexion à la base de donnée

- Sécurisation de l'accès à la base de données :

j'ai créé un utilisateur sur le Raspberry avec tous les droits pour obtenir un contrôle total sur le serveur MySQL. Cet utilisateur est nommé "AdminP". Ensuite, j'ai ajouté le compte utilisateur sur la base de données du technicien 2 . J'ai accordé à cet utilisateur tous les droits pour utiliser la base de données.

Survol des utilisateurs

Utilisateur	Client	Mot de passe	Privilèges globaux	«Grant»	Action
<input type="checkbox"/> AdminP	172.29.19.233	Oui	ALL PRIVILEGES	Oui	Changer les priviléges Exporter

- Connexion :

L'objet “bdd” (ligne 20) est déclaré de la classe BaseDeDonnees.h , ce qui permet de faire par la suite appelle à la méthode connexion().

Les paramètres représentent l'IP de la base de données, nom d'utilisateur, mot de passe, nom de la base de données.

Utilisation du programme a l'infinie

La déclaration de la variable "retourhaut" (ligne 35) permet de l'utiliser dans la boucle "while". Cette boucle permet la lecture continue des bracelets, car le programme ne se termine jamais, étant donné que les utilisateurs du parc aquatique continuent constamment de badger le lecteur. Ainsi, tant que la variable n'est pas égale à zéro, le code fonctionne en boucle.

PROGRAMME PRINCIPAL (suite) - Poiret Cédric

Projet informatique | Accès à l'attraction

Récupération du numéro de bracelet

S'il n'y a pas d'erreur à la connexion de la base de données, on initialise à l'aide la fonction gpioinitialise() (ligne 27) de la bibliothèque **wiringPI.h** pour la configuration de l'environnement GPIO .

J'ai déclaré **l'objet** de CLecteurRFID (ligne 34) de la classe CLecteurRFID.h pour pouvoir accéder aux méthodes.

On récupère le numéro du bracelet dans la variable "**decimalValue**" (ligne 37) grâce à la méthode lirebadge() , puis une condition de vérification est faite pour savoir si le bracelet lu est bien positif.

Utilisation de l'objet CRequete

Je déclare l'objet de la classe CRequete (ligne 41) avec en paramètre adresse de l'objet de la classe BaseDeDonnees pour effectuer mes requêtes.

Vérification de l'existence du badge :

La première chose à faire est de **vérifier** si le badge lu par le lecteur RFID est bien **valide**, donc on utilise la requête **verifierExistenceid()** (ligne 42) avec une **condition** (si la requête retourne true le badge est valide sinon invalide).

Test de la taille

On récupère la taille avec la méthode **recupererTaille()** (ligne 49).

On définit une condition pour savoir quelle taille minimum pour accéder à l'attraction, dans notre cas, j'ai fixé la taille minimum à 175 cm (ligne 52).

- Taille Correcte :

Si la taille est correcte, on déclare l'objet CLed avec le numéro du port GPIO (ligne 55). Puis on allume en **vert** puis avec un delay la led repasse en **rouge**.

- Taille incorrecte :

Si la taille est incorrecte, on ajoute +1 au nombre de refus du **visiteur** à l'aide de la méthode **ajouterAccesRefuser()**, et pour ajouter à l'historique des refus, on appelle la méthode **ajouterHistorique()** qui a ajouté à la base de données (le **numéro** du badge, date avec **l'heure**, **nom de l'attraction**).

CONCLUSION - Poiret Cédric

Projet informatique | Accès à l'attraction

Conclusion

Le test de ce programme a été concluant dans les différents cas d'utilisation du bracelet, ce qui puis permettre au technicien n°3 d'effectuer sa partie en réutilisant le code lecture du bracelet et de la requête permettant l'identification des visiteurs et pour le technicien n°2 d'établir les requêtes sur les informations reçues par la base de données.



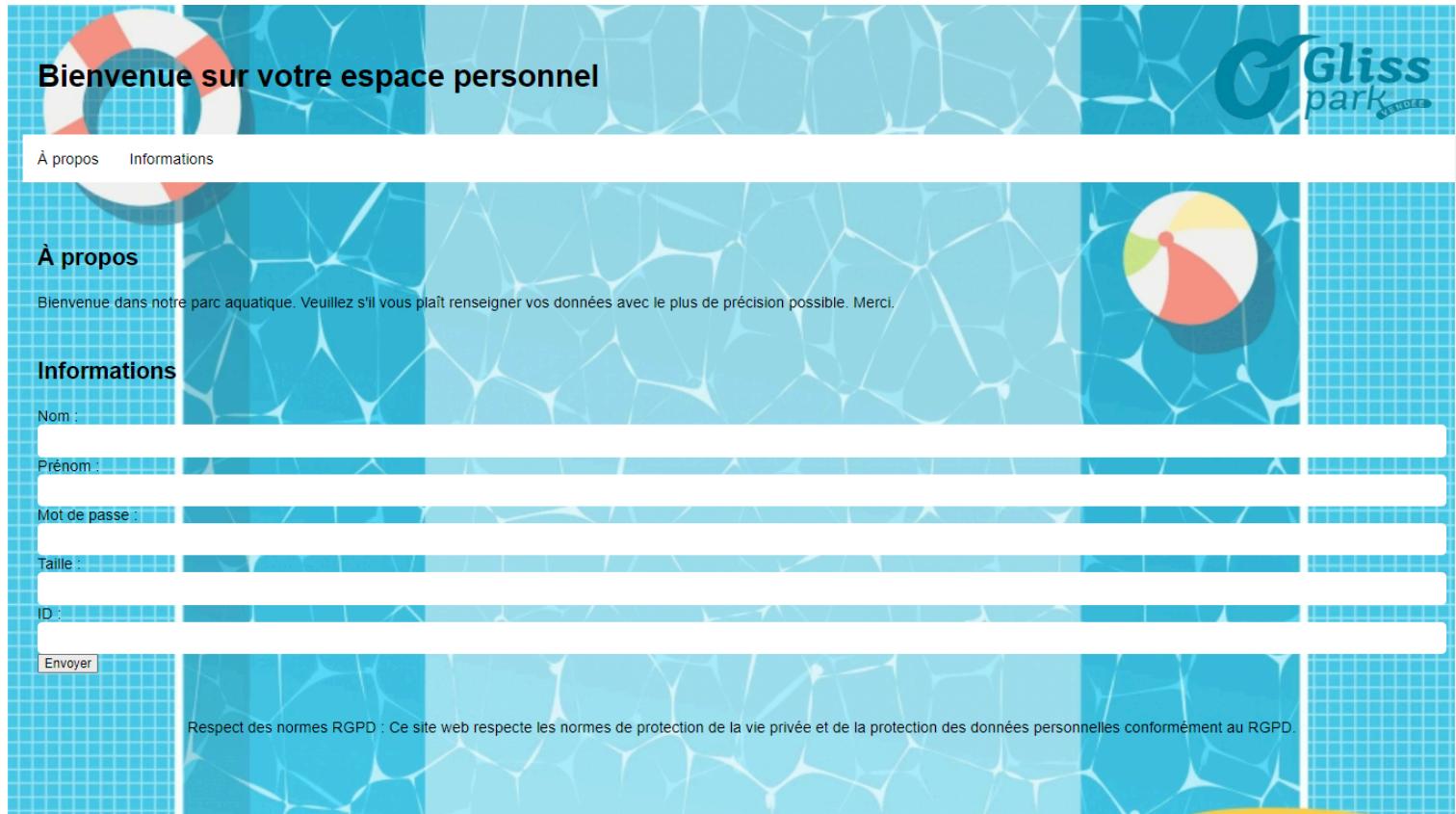
SITE INTERNET PARC (Laforgue Yann)

Projet informatique | WebSite

Je suis responsable de l'inscription des visiteurs à l'accueil du parc depuis une page HTML qui enverra les données à la base de données qui liera ceux- ci à l'identifiant du bracelet.

Suite à cette inscription je dois faire en sorte que les données du visiteurs puisse être réutilisé pour une connexion depuis une autre page HTML, pour que les visiteurs puissent consulter leurs données telle que les photos prise durant l'attraction, l'état de leurs casier (*ouvert, fermé*) et aussi les attractions auxquelles l'accès leurs a été refusé ainsi que l'heures à laquelle ils ont était refusé.

INSCRIPTION.PHP - Laforgé Yann



Cette page internet est mise en forme par un CSS qui lui est propre qui définit :

- La police d'écriture et le fond

```
body {  
    font-family: Arial, sans-serif;  
    background-color: #f7f7f7;  
}
```

- L'entête qui est séparé du reste contenant le titre et le logo du parc

```
header {  
    display: flex;  
    align-items: center;  
    justify-content: space-between;  
    padding: 1em;  
}
```

- Les zones de saisie une de type texte est un spécialement pour cacher le mot de passe, mais aussi la taille ainsi que leurs formes

```
form input[type="text"],
form input[type="password"],
form textarea {
padding: 0.5em;
font-size: 1em;
width: 100%;
box-sizing: border-box;
border-radius: 5px;
border: 1px solid #ccc;
}
```

- Et un pied de page pour préciser les norme RGPD de sécurité

```
footer {
text-align: center;
padding: 1em;

color: black;
}
```

Les zones de saisie sont tous de type texte sauf celle du mot de passe pour cacher les caractères et sont liées à un autre php (*traitement.php*) pour les stocker dans la base de données après le clique sur le bouton envoyer.

```
<form action="traitement.php" method="post">
<label for="nom">Nom :</label>
<input type="text" id="nom" name="nom"><br>

<label for="prenom">Prénom :</label>
<input type="text" id="prenom" name="prenom"><br>

<label for="motdepasse">Mot de passe :</label>
<input type="password" id="motdepasse" name="motdepasse"><br>

<label for="taille">Taille :</label>
<input type="text" id="taille" name="taille"><br>

<label for="ID">ID :</label>
<input type="text" id="ID" name="ID"><br>

<button type="submit" name="submit">Envoyer</button>
</form>
```

TRAITEMENT.PHP - Laforgé Yann

La connexion à la base de donnée se fait grâce à la fonction **mysqli_connect** et aux informations entrées en entrée de la page dans les variables pour pouvoir simplement changer les login sans avoir à chercher dans le code. Un message est envoyé en cas d'erreur de connexion grâce à la fonction **mysqli_connect_error**.

```
$host = 'localhost';
$username = 'root';
$password = '';
$database = 'parc_aquatique';

// Connexion à la base de données
$conn = mysqli_connect($host, $username, $password, $database);

// Vérifier la connexion
if (!$conn) {
    die("La connexion à la base de données a échoué : " . mysqli_connect_error());
}
```

Les informations précédemment entrées dans l'inscription sont ensuite grâce aux méthodes **post** et sont ensuite envoyées dans la base de donnée grâce à une **requête SQL** et à la fonction **mysqli_prepare** qui prépare la requête.

```
// Recuperer les données saisies
$nom = $_POST['nom'];
$prenom = $_POST['prenom'];
$motdepasse = $_POST['motdepasse'];
$taille = $_POST['taille'];
$ID = $_POST['ID'];

// Préparer la requête
$stmt = mysqli_prepare($conn, "INSERT INTO visiteurs (nom, prenoms, mot_de_passe, taille, ID) VALUES (?, ?, ?, ?, ?)");
```

Les paramètres sont ensuite liés.
Le 'ssssi' indique le type des variables, 's' pour string et 'i' pour int.

```
// Lier les paramètres à la requête préparée
mysqli_stmt_bind_param($stmt, "ssssi", $nom, $prenom, $motdepasse, $taille, $ID);
```

La requête est ensuite exécutée grâce à la fonction **mysqli_stmt_execute**.

CONNEXION WEB.PHP- Laforge Yann

Le visiteur peut ensuite se connecter à son espace personnel grâce à son nom et le mot de passe qu'il a renseigné sur la page d'inscription.



Les informations de cette page sont envoyées dans le formulaire php pour pouvoir les comparer grâce à la méthode **post**.

```
<form method="post" action="connexion.php">

    <div class="image" style="background-image: url('images/FondV2.jpg');">
        <div class="wrap-login100 p-t-30 p-b-50">
            <span class="titre">
                Connexion
            </span>
            <div class="zoneBlanche">

                <div class="zone" data-validate="nom">
                    <input class="text" type="text" id="nom" name="nom" placeholder="Nom">
                </div>

                <div class="zone" data-validate="password">
                    <input class="text" type="password" id="password" name="password" placeholder="Mot de passe">
                </div>
```

Cette page internet est aussi liée à un CSS pour la mise en forme qui permet de :

- Définir la taille et la forme des zones de saisie

```
* {
    margin: 0px;
    padding: 0px;
    box-sizing: border-box;
}

input {
    outline: none;
    border: none;
}
```

- Et la taille le forme et la couleur du bouton confirmer

```
.confirme {
    width: 100%;
    display: flex;
    justify-content: center;
}

.couleurbtn {
    font-family: Ubuntu-Bold;
    font-size: 18px;
    color: ■#fff;
    line-height: 1.2;

    min-width: 160px;
    height: 42px;
    border-radius: 21px;

    background: ■#d41872;
    background: -webkit-linear-gradient(left, ■#a445b2, ■#d41872, ■#fa4299);
}
```

Les données renseignées sont ensuite comparées à celles enregistrées dans la base de données grâce au formulaire php.

CONNEXION.PHP - Laforge Yann

Ce formulaire compare les données renseignées sont ensuite comparé à ceux enregister dans la base de données.

Il faut pour cela naturellement se connecter à la base de données grâce à la fonction **mysqli_connect**.

```
$host = 'localhost';
$username = 'root';
$password = '';
$database = 'parc_aquatique';

// Connexion a la base de donnees
$conn = mysqli_connect($host, $username, $password, $database);
```

Nous récupérons ensuite les données saisies grâce à la méthode **post**.

Nous faisons aussi attention à ce que si le mot de passe contient des caractères spéciaux qu'il soit pris en compte grâce à la fonction **mysql_real_escape_string**.

```
// Recuperer les informations du formulaire
$username = mysqli_real_escape_string($conn, $_POST['nom']);
$password = mysqli_real_escape_string($conn, $_POST['password']);
```

Après ceci nous comparons les données renseignées grâce à une **requête SQL**.

```
// Executer une requete SQL pour verifier les informations d'identification
$sql = "SELECT id FROM visiteurs WHERE nom = '$username' and mot_de_passe = '$password'";
$result = mysqli_query($conn, $sql);
```

Nous faisons une vérification que la requête ce soir bien effectué

```
// Vérifier si la requête SQL a réussi
if ($result === false) {
    // Si la requête a échoué, afficher un message d'erreur avec les informations de l'erreur
    echo 'Erreur SQL : ' . mysqli_error($conn);
} else {
```

Et si celle si c'est bien exécuter cela nous laisse deux options soit si les login sont correct permettra de rédiger sur la page personnel du client soit d'afficher qu'il y à une erreur dans les identifiants.

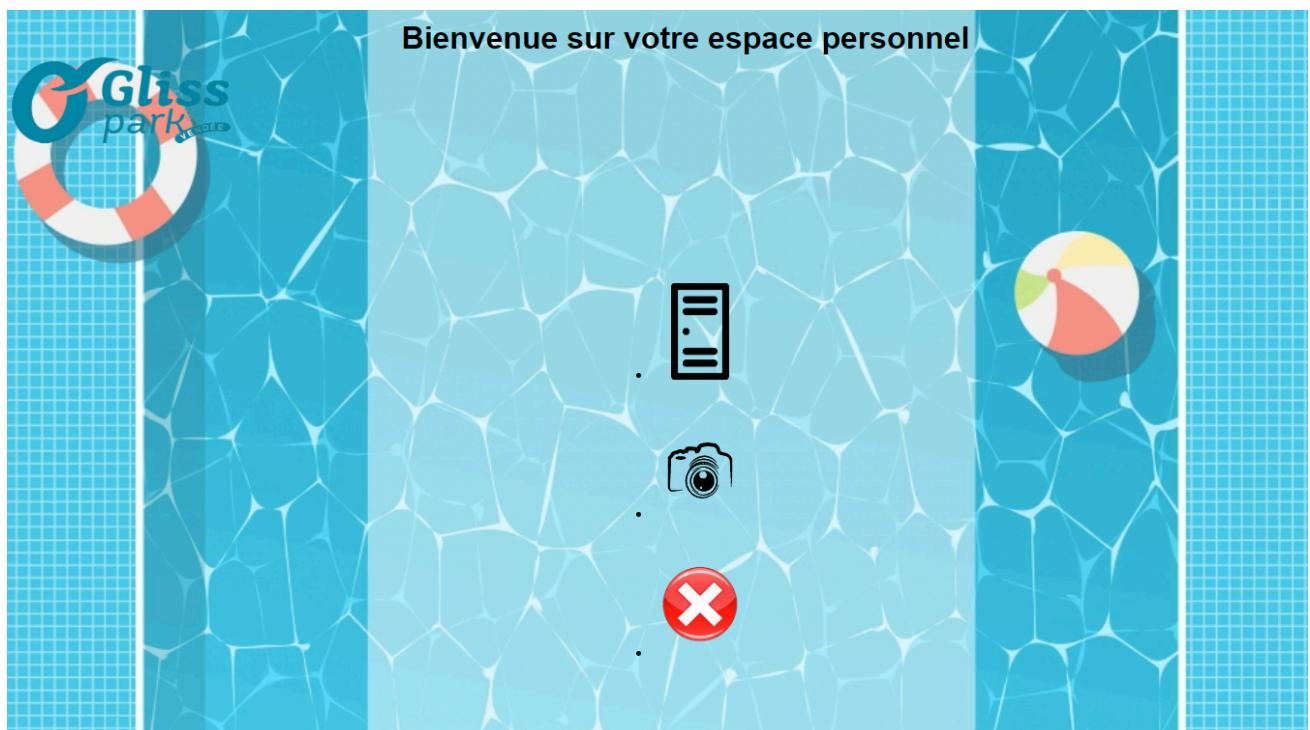
```
    } else {  
  
    // Sinon, verifier si les informations d'identification sont correctes  
    if (mysqli_num_rows($result) > 0) {  
        // Si les informations sont correctes, rediriger vers la page protegee  
        header('Location: http://localhost/Parc%20Aquatique/Interface.php');  
  
    } else {  
        // Sinon, afficher un message d'erreur  
        echo 'Nom d\'utilisateur ou mot de passe incorrect';  
    }  
}
```

INTERFACE.PHP- Laforge Yann

Si la connexion est bien effectuée sur la page précédente, le visiteur aura accès à son espace personnel.

```
<?php
session_start();
if (isset($_SESSION['nom'])) {
    $nom = $_SESSION['nom'];
    echo "<h1>Bienvenue Mr $nom!</h1>";
} else {
    echo "<h1>Veuillez saisir votre nom dans le formulaire précédent</h1>";
}
?>
```

Sur cette espace le visiteur aura accès à différentes informations telle que l'état de son casier les photos prises durant les activités et les activités auxquelles l'entrée lui a été refusée.



CONCLUSION - Laforge Yann

Mon rôle dans ce projet a donc permis d'identifier les visiteurs de stocker toutes les données de manière fiable et sécurisée telles que les différentes données envoyées par les deux autres techniciens comme les attractions refusées et les photos.

Projet informatique | Accès au Casier

Présentation

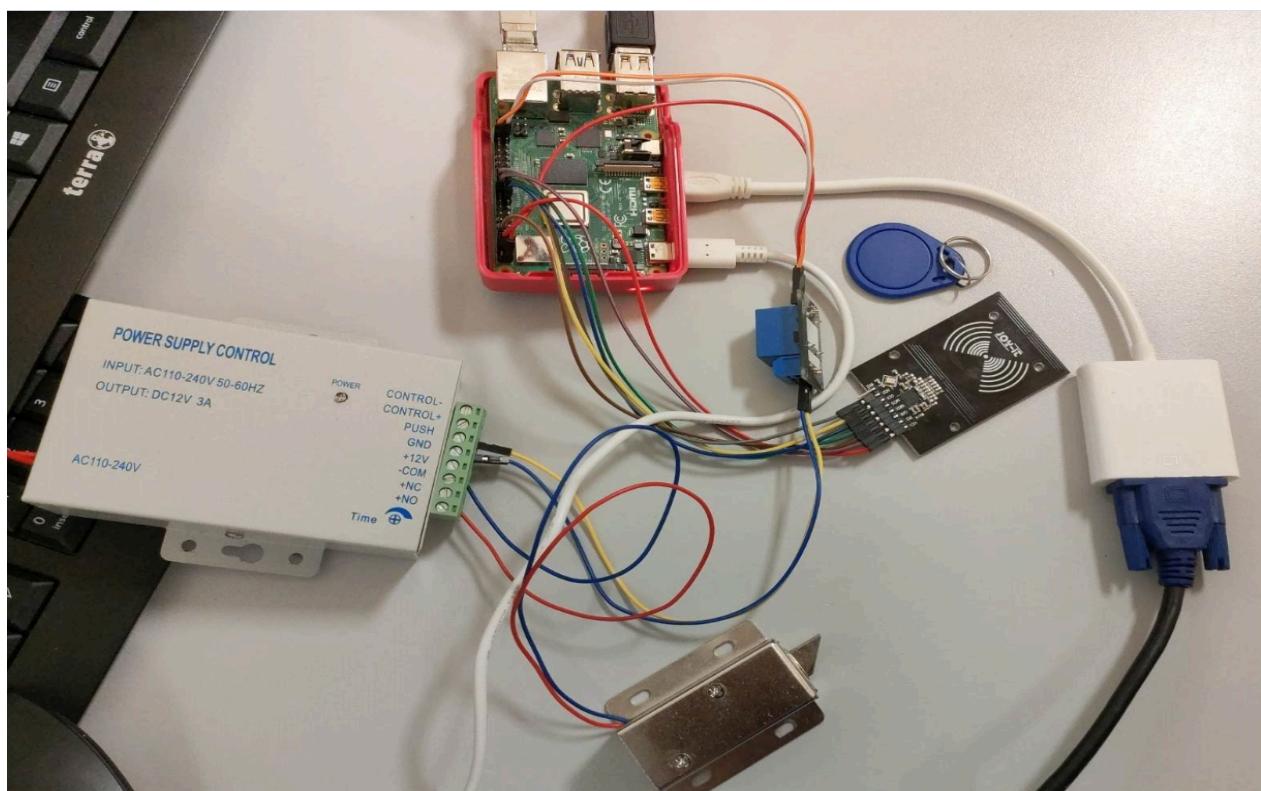
Ce logiciel permet de contrôler les accès au Casier afin de stocker les biens en toute sécurité.

Fonctionnement

Ce système informatique permet de sécuriser les biens des visiteurs et qui ont accès à leurs casiers grâce aux bracelets RFID.

Le logiciel permet donc :

- Identifier les droits des visiteurs avec son bracelet.
- D'archiver les refus de l'accès à la base de données (date,nom de l'attraction).



Projet informatique | Accès au Casier

Code source Clecteur.cpp

```
#include "ClecteurRFID.h" // Inclusion du fichier d'en-tête "ClecteurRFID.h"
#include <iostream> // Inclusion de la bibliothèque iostream
#include <sstream> // Inclusion de la bibliothèque sstream
#include <math.h> // Inclusion de la bibliothèque math.h
#include "pigpio.h" // Inclusion du fichier d'en-tête "pigpio.h"
#include <unistd.h> // Inclusion de la bibliothèque unistd.h
#include <wiringPi.h> // Inclusion de la bibliothèque wiringPi

ClecteurRFID::ClecteurRFID()
{
    // Constructeur
}

ClecteurRFID::~ClecteurRFID()
{
    // Destructeur
}

void ClecteurRFID::fermerportspi()
{
    gpioTerminate(); // Terminaison de la bibliothèque GPIO
}

long long int ClecteurRFID::lirebadge()
{
    long long int decimalValue = 0; // Variable pour stocker la valeur décimale du badge
    int rep = 1; // Variable pour contrôler la boucle
    mfrc522.PCD_Init(); // Initialisation du lecteur RFID
    while (rep == 1) {
        if (mfrc522.isNewCardPresent() == 0) { // Détection d'un badge RFID au niveau du lecteur
            // std::cout << "Nouveau badge détecté..." << std::endl;
            delay(1000); // Attente de 1 seconde
            if (mfrc522.PICC_ReadCardSerial()); // Lecture de la carte RFID
            uint8_t nuidPICC[5]; // Tableau pour stocker l'identifiant du badge
            for (int i = 11; i < 16; ++i) {
                nuidPICC[i - 11] = mfrc522.uid.uidByte[i]; // Copie de l'identifiant du badge dans le tableau
            }

            printf("\n");
            for (int i = 0; i < 4; ++i) {
                decimalValue += nuidPICC[i] << (8 * i); // Conversion de l'identifiant du badge en valeur décimale
            }
            return decimalValue; // Retour de la valeur décimale du badge
        }
    }
}
```

Ce code en langage c++ (repris par le technicien 1) sert à initialiser l'équipement lecteur RFID afin de lire et afficher le numéro du badge/bracelet qui est en RFID, pour pouvoir utiliser dans le contexte du casier. Car dans le contexte du casier, faudra identifier le numéro du bracelet pour savoir s'il a les droits d'ouverture du casier.

La classe ClecteurRFID.h

```
#ifndef CLECTEURRFID_H
#define CLECTEURRFID_H

#include "CMFRC522.h" // Inclusion du fichier d'en-tête "CMFRC522.h"
#include <pigpio.h> // Inclusion du fichier d'en-tête "pigpio.h"

class ClecteurRFID
{
public:
    ClecteurRFID(); // Constructeur
    ~ClecteurRFID(); // Destructeur
    void fermerportspi(); // Méthode pour fermer le port SPI
    long long int lirebadge(); // Méthode pour lire le badge RFID

private:
    CMFRC522 mfr522; // Instance de la classe CMFRC522 pour interagir avec le lecteur RFID
};

#endif // CLECTEURRFID_H
```

Ce code correspond à une classe qui s'associe avec le fichier précédent ClecteurRFID.cpp. Cette classe permet de désigner les méthodes pour fermer le port SPI et la lecture du badge RFID. La méthode “lirebadge()” est utilisée pour lire le badge RFID et renvoyer un identifiant sous forme d'entier long long int.

Projet informatique | Accès au Casier

Code source Main.cpp

```
#include <iostream> // Inclusion de la bibliothèque iostream
#include <unistd.h> // Inclusion de la bibliothèque unistd.h
#include "ClecteurRFID.h" // Inclusion du fichier d'en-tête "ClecteurRFID.h"
#include <wiringPi.h> // Inclusion de la bibliothèque wiringPi
#include "BaseDeDonnees.h" // Inclusion du fichier d'en-tête "BaseDeDonnees.h"
#include "requete.h" // Inclusion du fichier d'en-tête "requete.h"
#include "CMFRC522.h" // Inclusion du fichier d'en-tête "CMFRC522.h"

using namespace std;

void delay(int ms) {
#ifdef WIN32
    Sleep(ms);
#else
    usleep(ms * 1000);
#endif
}

int main(int argc, char* argv[]) {
    BaseDeDonnees bdd; // Instance de la classe BaseDeDonnees
    if (!bdd.connexion("172.29.21.52", "utilisateur", "root", "parc")) { // Connexion à la base de données
        cerr << "Erreur de connexion à la base de données." << endl;
        return 1;
    }
    else {
        if (gpioInitialise() < 0) { // Initialisation du port SPI
            cerr << "Port SPI non initialisé." << endl;
        }
        else {
            ClecteurRFID monlecteur; // Instance de la classe ClecteurRFID
            int retourhaut = 1; // Variable pour contrôler la boucle
            while (retourhaut != 0) {
                long long int decimalValue = monlecteur.lirebadge(); // Lecture du badge RFID
            }
        }
    }
}
```

Ce code source Main.cpp est mon programme principale qui permet dans un premier temps de se connecter à la base de donnée (**actuellement en cours de développement**) en utilisant les paramètres spécifiés (adresse IP, nom d'utilisateur, mot de passe et nom de la base de données), puis initialiser le port SPI pour pouvoir lire le bracelet RFID. En cas d'échec, un message d'erreur sera affiché.

La méthode “verifierExistenceId()” permet de vérifier si l'identifiant existe dans la base de données. Le résultat est stocké dans la variable booléenne “existe”.

Si l'identifiant existe, un message est affiché indiquant sa présence dans la base de données. Sinon, un message d'erreur sera affiché.

```

    if (decimalValue != 0) {
        printf("Numéro de badge converti en base 10: %lli\n", decimalValue);
        Requete req(&bdd); // Instance de la classe Requete avec la base de données en paramètre
        bool existe = req.verifierExistenceId(decimalValue); // Vérification de l'existence de l'ID dans la base de données

        std::cout << "la valeur du booléen est : " << existe << std::endl;
        if (existe == true) {
            std::cout << "L'ID " << decimalValue << " existe dans la base de données." << std::endl;
        }
        else {
            cerr << "L'ID " << decimalValue << " n'existe pas dans la base de données." << endl;
            // break;
        }
    }
    delay(1000); // Attente de 1 seconde
    retourhaut = 1;
}
monlecteur.fermerportspi(); // Fermeture du port SPI
}

return 0;
}

```

C-BaseDeDonnées

VASSEUR Pierre

Projet informatique | Accès au Casier

Code source BaseDeDonnées.cpp

```

#include "BaseDeDonnees.h" // Inclusion du fichier d'en-tête "BaseDeDonnees.h"
#include <iostream> // Inclusion de la bibliothèque iostream
#include "requete.h" // Inclusion du fichier d'en-tête "requete.h"

BaseDeDonnees::BaseDeDonnees() {}

BaseDeDonnees::~BaseDeDonnees() {}

bool BaseDeDonnees::connexion(char* server, char* user, char* password, char* database) {
    //printf(Base;connexion())
    mysql_init(&mysql); // Initialisation de la structure MYSQL
    maBase = mysql_real_connect(&mysql, server, user, password, database, 0, 0, 0); // Connexion à la base de données

    if (maBase == NULL) {
        return false;
    }
    return true;
}

bool BaseDeDonnees::deconnexion() {
    mysql_close(maBase); // Fermeture de la connexion à la base de données
    return true;
}

bool BaseDeDonnees::envoyerRequete(char* requete) {
    int query_state;
    query_state = mysql_query(&mysql, requete); // Envoi de la requête SQL à la base de données
    if (query_state != 0) {
        return false;
    }
    res = mysql_store_result(maBase); // Stockage du résultat de la requête
    return true;
}

MYSQL_RES* BaseDeDonnees::GetResponse() {
    return res; // Renvoie le résultat de la requête
}

```

La BaseDeDonnées.cpp permet d'être en relation avec la classe BaseDeDonnées.h dans le but initialisé la structure MYSQL pour pouvoir se connecter à la base de données ainsi d'émettre et recevoir des requêtes SQL

à la base de données en utilisant la fonction “mysql_query”. Si la connexion échoue, elle renvoie “false”, sinon elle renvoie “true”.

La méthode “envoyerRequete()” est définie avec le paramètre “requete”.

Si la requête échoue, elle renvoie “false”, sinon elle stocke le résultat de la requête dans la variable “res” à l'aide de la fonction “mysql_store_result” et renvoie “true”.

La méthode “GetResponse()” est définie et renvoie le résultat de la requête stocké dans la variable “res”.

C-BaseDeDonnées (*suite*)

VASSEUR Pierre

Projet informatique | Accès au Casier

La classe BaseDeDonnées.h

```
#ifndef BASE_DE_DONNEES_H
#define BASE_DE_DONNEES_H

#include <mysql/mysql.h> // Inclusion du fichier d'en-tête "mysql.h"
#include <string> // Inclusion de la bibliothèque string

class BaseDeDonnees {
private:
    MYSQL mysql; // Structure MYSQL pour la gestion de la base de données
    MYSQL* maBase; // Pointeur vers la connexion à la base de données
    MYSQL_RES* res; // Pointeur vers le résultat d'une requête

public:
    BaseDeDonnees(); // Constructeur
    ~BaseDeDonnees(); // Destructeur
    bool connexion(char* server, char* user, char* password, char* database); // Méthode pour établir une connexion à la base de données
    bool deconnexion(); // Méthode pour se déconnecter de la base de données
    bool envoyerRequete(char* requete); // Méthode pour envoyer une requête SQL à la base de données
    MYSQL_RES* GetResponse(); // Méthode pour obtenir le résultat d'une requête
};

#endif // BASE_DE_DONNEES_H
```

La classe BaseDeDonnées.h permet d'être en relation avec la BaseDeDonnées.cpp pour pouvoir inclure quelque méthode afin de communiquer entre l'envoie de la requête vers la base de données et recevoir le résultat de la requête.

Projet informatique | Accès au Casier

Code source requete.cpp

```
#include "requete.h" // Inclusion du fichier d'en-tête "requete.h"
#include "BaseDeDonnees.h" // Inclusion du fichier d'en-tête "BaseDeDonnees.h"
#include <iostream> // Inclusion de la bibliothèque iostream
#include "CMFRC522.h" // Inclusion du fichier d'en-tête "CMFRC522.h"
#include <chrono> // Inclusion de la bibliothèque chrono

Requete::Requete(BaseDeDonnees* bdd) {
    m_bdd = bdd; // Assignation de l'objet BaseDeDonnees passé en paramètre
}

Requete::~Requete() {}

bool Requete::verifierExistenceId(long long int numerobadge) {
    char req[500];
    sprintf(req, "SELECT ID FROM visiteur WHERE ID=%lld", numerobadge); // Création de la requête SQL pour vérifier l'existence de l'ID
    //dans la table "visiteur"

    printf("La requête est %s\n", req);

    if (m_bdd->envoyerRequette(req)) { // Envoi de la requête à la base de données
        MYSQL_RES* resultat = REPONSE(); // Récupération du résultat de la requête
        if (resultat != NULL) {
            MYSQL_ROW row = mysql_fetch_row(resultat); // Récupération de la première ligne du résultat
            mysql_free_result(resultat); // Libération de la mémoire occupée par le résultat
            if (row != NULL) {
                return true; // L'ID existe dans la base de données
            }
        }
    }
    return false; // L'ID n'existe pas dans la base de données
}
```

Le requete.cpp permet de paramétriser l'objet pour la base de donnée puis créer la requête SQL pour vérifier si L'ID donc le numéro du bracelet est reconnu ou pas dans la table “visiteur” qui se trouve dans la base de donnée. Mais si l'ID n'est pas reconnu , la serrure du casier ne sera pas actionnée.

La classe requete.h

```
#ifndef REQUETE_H
#define REQUETE_H

#include <string> // Inclusion de la bibliothèque string
#include <vector> // Inclusion de la bibliothèque vector
#include "BaseDeDonnees.h" // Inclusion du fichier d'en-tête "BaseDeDonnees.h"

class Requete {
private:
    BaseDeDonnees* m_bdd; // Pointeur vers l'objet BaseDeDonnees

public:
    Requete(BaseDeDonnees* bdd); // Constructeur prenant en paramètre un objet BaseDeDonnees
    ~Requete(); // Destructeur
    bool verifierExistenceId(long long int numerobadge); // Méthode pour vérifier l'existence d'un ID dans la base de données

    MYSQL_RES* REPONSE(); // Méthode pour obtenir le résultat d'une requête
};

#endif // REQUETE_H
```

La classe requete.h sert d'inclure les méthodes suivantes :

-La méthode “Requete::Requete” est un constructeur qui prend en paramètre un pointeur vers un objet “BaseDeDonnees”. Cela permet d’établir une connexion à la base de données et d’utiliser cet objet pour exécuter les requêtes.

-La méthode “Requete::verifierExistenceld” est utilisée pour vérifier si un ID spécifique existe dans la base de données. Elle prend en paramètre un entier long long int qui représente l’ID à vérifier.

-A l’intérieur de cette méthode, une requête SQL est construite pour effectuer la vérification. La requête est envoyée à la base de données en utilisant l’objet “BaseDeDonnees” associé à la classe “Requete”.

- Si la requête est exécutée avec succès, le résultat est récupéré et analysé. Si une ligne est retournée, cela signifie que l’ID existe dans la base de données, et la méthode renvoie “true”. Sinon, elle renvoie “false”.

-Et la méthode “Requete::REPONSE” est utilisée pour récupérer le résultat d’une requête exécutée sur la base de données.

C-Serrure

VASSEUR Pierre

Projet informatique | Accès au Casier

Le code source main.cpp

```
#include "lecteurRFID.h" // Inclusion du fichier d'en-tête "lecteurRFID.h"
#include "requete.h" // Inclusion du fichier d'en-tête "requete.h"
#include <stdio.h> // Inclusion de la bibliothèque standard d'entrée/sortie
#include <stdlib.h> // Inclusion de la bibliothèque standard
#include <time.h> // Inclusion de la bibliothèque time
#include <stdbool.h> // Inclusion de la bibliothèque standard bool
#include <wiringPi.h> // Inclusion de la bibliothèque WiringPi
#include "serrure.h" // Inclusion du fichier d'en-tête "serrure.h"
#include <iostream> // Inclusion de la bibliothèque iostream
```

```
srand(time(NULL)); // Initialisation du générateur de nombres aléatoires
Serrure serrure; // Création d'un objet Serrure
```

```
int random_number = rand() % 2; // Génération dun nombre aléatoire entre 0 et 1
if (random_number == 0) {
    serrure.unlock(); // déverrouiller la serrure
```

J’ai créé ce deuxième code principale uniquement pour l’interaction de la serrure en attendant la fin du développement de mon premier code principal pour qui puisse interagir avec les autres équipements.

Projet informatique | Accès au Casier

Code source serrure.cpp

```
#include "serrure.h"

Serrure::Serrure() {
    wiringPiSetup(); // initialiser la bibliothèque wiringPi
    pinMode(RELAY_PIN, OUTPUT); // configurer la broche GPIO comme une sortie
    relay_pin = RELAY_PIN;
}

void Serrure::unlock() {
    printf("La serrure est déverrouillée.\n");
    digitalWrite(relay_pin, HIGH); // activer le module relais (serrure déverrouillée)
    delay(LOCK_TIME); // attendre
    digitalWrite(relay_pin, LOW); // désactiver le module relais (serrure verrouillée)
    printf("La serrure est verrouillée.\n");
}
```

Le constructeur de la classe “serrure” est définie. Il initialise la bibliothèque wiringPi en appelant la fonction “wiringPiSetup()”. Ensuite, il configure la broche GPIO (relay_pin) utilisée pour contrôler le module relais en tant que sortie en utilisant la fonction “pinMode()”. La constante “RELAY_PIN” est utilisée pour spécifier le numéro de broche GPIO. Finalement, la variable membre “relay_pin” est initialisée avec la valeur de “RELAY_PIN”.

La méthode “unlock()” est définie. Elle est utilisée pour déverrouiller la serrure. Elle affiche un message indiquant que la serrure est déverrouillée en utilisant la fonction “digitalWrite()” pour mettre la broche GPIO en état haut (HIGH), ce qui déverrouille la serrure. Elle attend pendant une durée spécifiée par la constante “LOCK_TIME” en utilisant la fonction “delay()”. Après cette attente, elle désactive le module relais en mettant la broche GPIO en état bas (LOW), ce qui verrouille la serrure. Enfin, elle affichera un message indiquant que la serrure est verrouillée.

Projet informatique | Accès au Casier

Classe serrure.h

```
ifndef SERRURE_H
#define SERRURE_H

#include <wiringPi.h>
#include <stdio.h>

#define RELAY_PIN 29 // numéro de la broche GPIO utilisée pour le module relais
#define LOCK_TIME 2000 // durée de verrouillage/déverrouillage (en millisecondes)

class Serrure {
public:
    Serrure(); // constructeur
    void unlock(); // méthode pour déverrouiller la serrure

    //void lock();

private:
    int relay_pin; // numéro de la broche GPIO utilisée pour le module relais
};

#endif
```

Cette classe s'associe au code source serrure.cpp pour le déroulement de la méthode déverrouiller de la serrure.

Projet informatique | Prise De Photo

Présentation

Ce logiciel permet de prendre des photos au moment où le détecteur de présence détecte un obstacle.

Fonctionnement

Ce système informatique permet de prendre des photos des visiteurs avec leurs ID ainsi que la date et heure. Visionnement possible grâce en les stockant sur la base de données.

Le logiciel permet donc :

- Prise de Photo des visiteurs.
- Synchroniser avec la base de données.



Projet informatique | Prise De Photo

Code source main.cpp

```

#include <stdio.h> // Inclusion de la bibliothèque standard d'entrée/sortie
#include <stdlib.h> // Inclusion de la bibliothèque standard
#include <wiringPi.h> // Inclusion de la bibliothèque WiringPi

#define TRIG_PIN 15 // Définition de la broche TRIG
#define ECHO_PIN 16 // Définition de la broche ECHO

int main() {
    wiringPiSetup(); // Initialisation de la bibliothèque WiringPi
    pinMode(TRIG_PIN, OUTPUT); // Configuration de la broche TRIG en sortie
    pinMode(ECHO_PIN, INPUT); // Configuration de la broche ECHO en entrée

    while (1) {
        digitalWrite(TRIG_PIN, LOW); // Mise à LOW de la broche TRIG
        delayMicroseconds(2); // Attente de 2 microsecondes
        digitalWrite(TRIG_PIN, HIGH); // Mise à HIGH de la broche TRIG
        delayMicroseconds(10); // Attente de 10 microsecondes
        digitalWrite(TRIG_PIN, LOW); // Mise à LOW de la broche TRIG

        while (digitalRead(ECHO_PIN) == LOW); // Attente du signal HIGH sur la broche ECHO
        long start_time = micros(); // Enregistrement du temps de début

        while (digitalRead(ECHO_PIN) == HIGH); // Attente du signal LOW sur la broche ECHO
        long end_time = micros(); // Enregistrement du temps de fin

        long travel_time = end_time - start_time; // Calcul du temps de parcours
        int distance = travel_time / 58; // Calcul de la distance en centimètres

        printf("Distance: %dcm\n", distance); // Affichage de la distance

        if (distance < 30) {
            char command[] = "gphoto2 --capture-image-and-download --filename /home/pi/Images/photo.jpg"; // Commande pour prendre une photo
            system(command); // Exécution de la commande
            break;
        }

        delay(1000); // Attente de 1 seconde avant de recommencer la boucle
    }
    return 0;
}

```

Ce code utilise la bibliothèque WiringPi pour mesurer la distance à l'aide d'un capteur à ultrasons connecté aux broches TRIG_PIN et ECHO_PIN.
La boucle principale effectue les opérations suivantes :

- 1 - Il envoie une impulsion ultrasonique en mettant la broche TRIG_PIN à HIGH puis à LOW.
- 2 - Attente de la réception du signal sur la broche ECHO_PIN.
- 3 - Mesure du temps de parcours de l'onde ultrasonique.
- 4 - Calcul de la distance en centimètre en ultrasonique.
- 5 - Affichage de la distance depuis le terminal.
- 6 - Si la distance est inférieure à 30 cm, la commande “gphoto2” est exécutée pour prendre une photo.
- 7 - Attente d'une seconde avant de recommencer la boucle.

Installation GPHOTO2 :

Pour ce faire , on se rend dans le terminal du raspberry , puis on va mettre à jour la liste de packages en exécutant les deux commandes suivantes :

- sudo apt-get update
- sudo apt-get upgrade.

Puisque le système du raspberry est maintenant à jour, il faut télécharger les packages dont nous aurons besoin pour compiler le logiciel GPHOTO2 avec la commande suivante :

- sudo apt-get install git make autoconf libltdl-dev libusb-dev libexif-dev libpopt-dev libxml2-dev libjpeg-dev libgd-dev gettext autopoint.

Une fois que tous les packages auront terminé l'installation, nous pourrons procéder à la récupération du code source de libgphoto2 et à sa compilation. Libgphoto2 est la bibliothèque sur laquelle gphoto2 est construite. Pour récupérer sa dernière version depuis github , on peut exécuter la commande suivante : - git clone <https://github.com/gphoto/libgphoto2.git>

Pour la compilation , va falloir exécuter les commandes suivantes et pouvoir installer le script makefile :

- cd ~/libgphoto2
- autoreconf --install --symlink
- ./configure
- make
- sudo make install

Ensuite pour cloner le code source de GPHOTO2 qui est sur le raspberry , faudra exécuter les deux commandes suivantes :

- cd ~
- git clone <https://github.com/gphoto/gphoto2.git>

Pour configurer le logiciel GPHOTO2 pour la compilation , on va devoir exécuter les commandes suivantes :

- cd ~/gphoto2
- autoreconf --install --symlink
- ./configure
- make
- sudo make install

Installation GPHOTO2 (suite) :

Pour vérifier GPHOTO2 trouve bien la bibliothèque , faut exécuter cette ligne de commande : - sudo nano /etc/ld.so.conf.d/libc.conf

Ensuite dans le fichier trouvé, le texte va apparaître à l'écran

"# libc default configuration"
"/usr/local/lib".

Puis on actualisera le cache de configuration avec la commande suivante :
- sudo ldconfig.

Et pour finir , pour générer le fichier de base de donnée matérielle pour **udev** avec la commande suivante :

- /usr/local/lib/libgphoto2/print-camera-list hwdb | sudo tee
/etc/udev/hwdb.d/20-gphoto.hwdb

Pour connaitre si GPHOTO2 est bien installer , on peut connaitre sa version avec cette ligne de commande : - gphoto2 --version

Fiche de recette Contexte Casier :

		Projet				
		Fiche de test				
*Numéro :	1			*Rédacteur :	Vasseur Pierre	*Date de création : 30/01/2023
*Titre du test :	Prise de photo sur une attraction					
*Description :	Prise de photo depuis une attraction					
Procédure de vérification de Conformité Technique				Résultat		
N°	Description	Résultat Attendu	Méthode de contrôle (affichage, log, script)	OK	Niveau d'anomalie	Commentaire
1	Le visiteur se présente à l'attraction	Identifiant requise au portique	Affichage Voyant			
2	Déclenchement des détecteurs à Ultrason	Visiteur détecter	Script Python			
3	Pise de la photo	Fichier de la photo	Visionner le fichier			
4	Transfert du Fichier de la photo sur le serveur	Création de l'enregistrement dans la BDD (identifiant du visiteur et le fichier de la photo)	La date, heure et L'ID			
Conclusion du test						
Date de fin:			Statut :			
Liste des N° d'anomalie :			Anomalies corrigées :			
Commentaire :						

Fiche de recette Contexte Photo :

		Projet				
		Fiche de test				
*Numéro :	2			*Rédacteur :	Vasseur Pierre	*Date de création : 31/01/2023
*Titre du test :	Programme d'heure ouverture et fermeture					
*Description :	Enregistrement des heures ouverture et fermeture du casier					
Procédure de vérification de Conformité Technique				Résultat		
N°	Description	Résultat Attendu	Méthode de contrôle (affichage, log, script)	OK	Niveau d'anomalie	Commentaire
1	Le visiteur se présente au casier RFID	Visiteur Badge	Affichage Voyant			
2	Ouverture du casier RFID	Enregistrement date et heure Ouverture	Affichage écran			
3	Fermeture du casier RFID	Enregistrement date et heure Fermeture	Affichage écran			
Conclusion du test						
Date de fin:			Statut :			
Liste des N° d'anomalie :			Anomalies corrigées :			
Commentaire :						