

SNIR

Communication Grandmont

Lycée Grandmont

2023 – 2024

Timéo Le Bronec, Louis Capdeville, Gaëtan Rousseau

Sommaire –

- 1- Présentation du projet
 - 1.1- Présentation
 - 1.2- Choix matériels et logiciels
 - 1.3- Cahier des charges
 - 1.4- Analyse UML
 - 1.5- Recette globale
- I- Installation du Raspberry Pi, mise en place de la base de données et réalisation de l'exécutable – Capdeville Louis**
 - 1- Installation et configuration du Raspberry Pi
 - 1.1- Configuration de l'OS du Raspberry Pi
 - 1.2- Installation d'un serveur Apache2
 - 1.3- Installation de MariaDB et PHPMyAdmin
 - 2- Mise en place de la base de données
 - 3- Recherche de solution pour afficher
 - 4- Programmation de la classe tBDD et création de l'exécutable
 - 4.1- La classe tBDD
 - 4.2 - Le main et l'exécutable
 - 5- Les recettes et tests
 - 5.1- Tests unitaires
 - 5.2- Tests d'intégrations
 - 5.3- Recettes
 - 6- Problèmes rencontrés et solutions
- II- Envoi de fichiers et responsive – Gaëtan Rousseau**
 - 1- Envoi de Fichiers
 - 1.1- Création
 - 2- Gestion du responsive
 - 2.1- Conception
 - 3- Cahier de recette
 - 3.1- Fichiers
- III- Envoi de messages et authentification – Timéo Le Bronec**
 - 1- Envoi de messages
 - 1.1- Conception
 - 2- Authentification
 - 2.1- Conception
 - 3- Cahier de recette
 - 3.1- Messages
 - 3.2- Authentification

3- Présentation du projet

1.1- Présentation

Le DDFPT a demandé un système pour présenter des informations régulières ou importantes à l'entrée du bâtiment I du lycée Grandmont. Le but de ce système est d'afficher des messages, des images ou des présentations PowerPoint à une date et heure choisies comme par exemple un message de bienvenue, les professeurs absents ou le menu de la cantine.

1.2- Choix matériels et logiciels

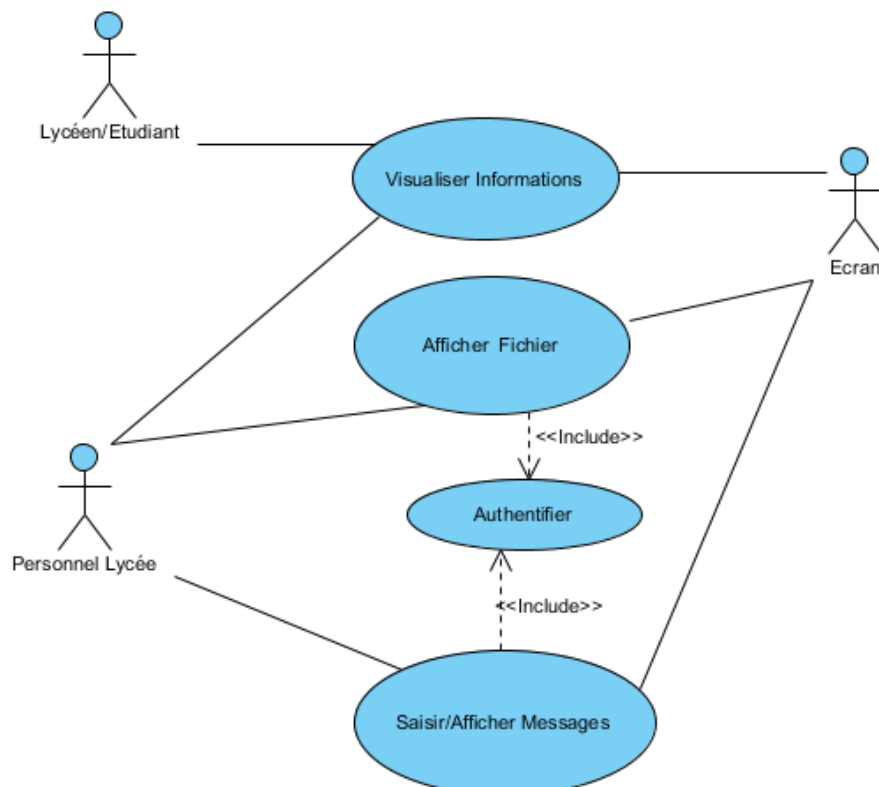
Pour ce projet, on nous fournit comme matériel un écran et un Raspberry Pi 3, c'est sur ce Raspberry Pi que seront hébergés la base de données MySQL, le site web PHP sur lequel les utilisateurs enverront leurs messages ou fichiers ainsi que l'application qui servira à les afficher qui, elle, sera en C++.

1.3- Cahier des charges

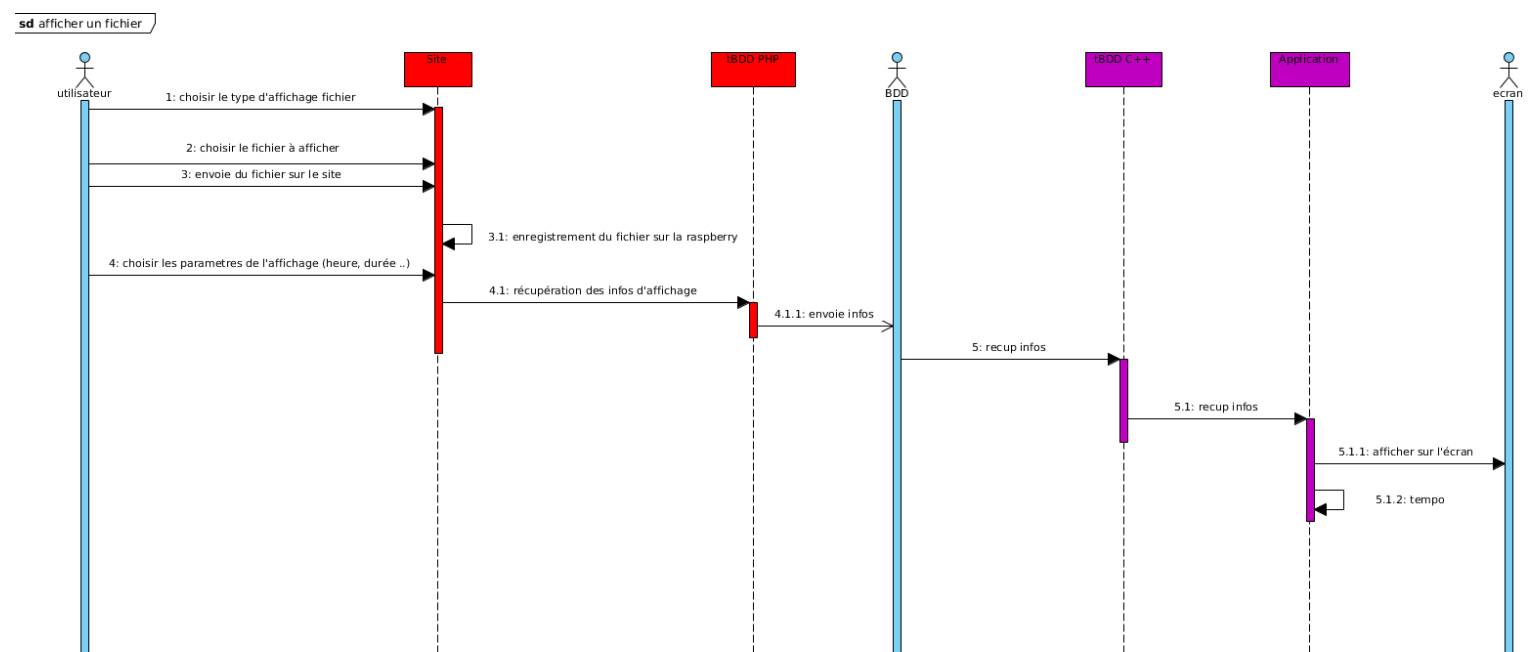
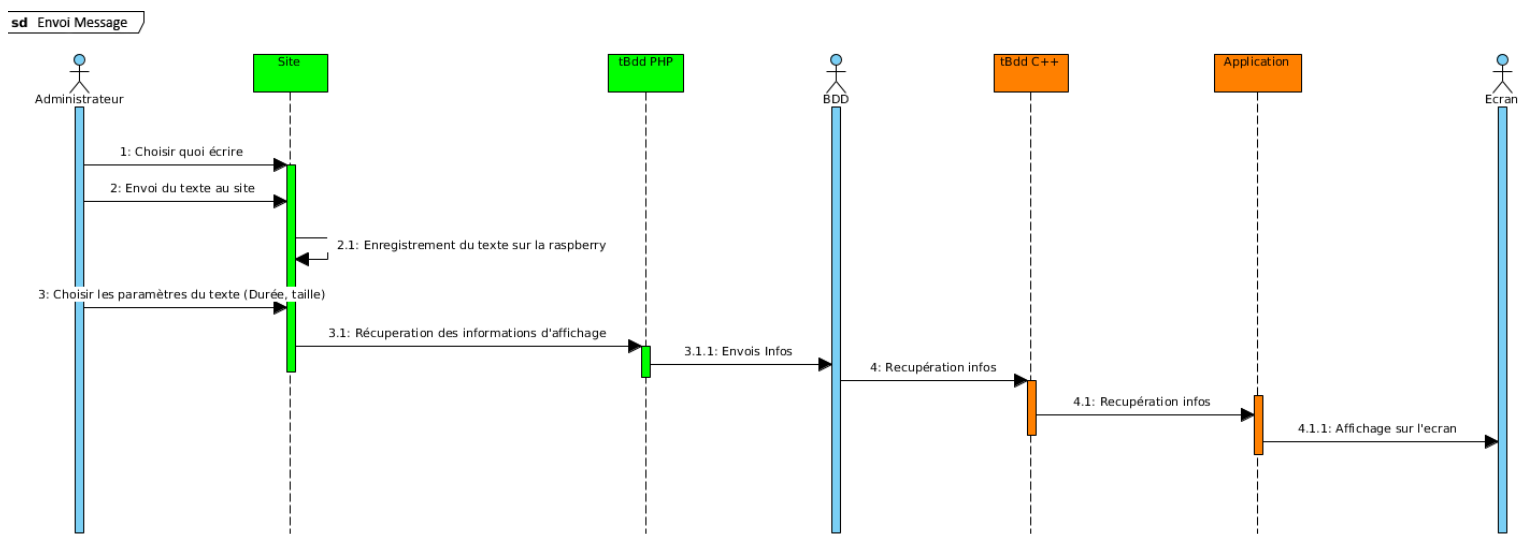
Le système doit être capable d'afficher des messages, images ou présentations à une heure choisie par l'utilisateur.

1.4- Analyse UML

Diagramme de cas d'utilisation



Diagrammes de séquence



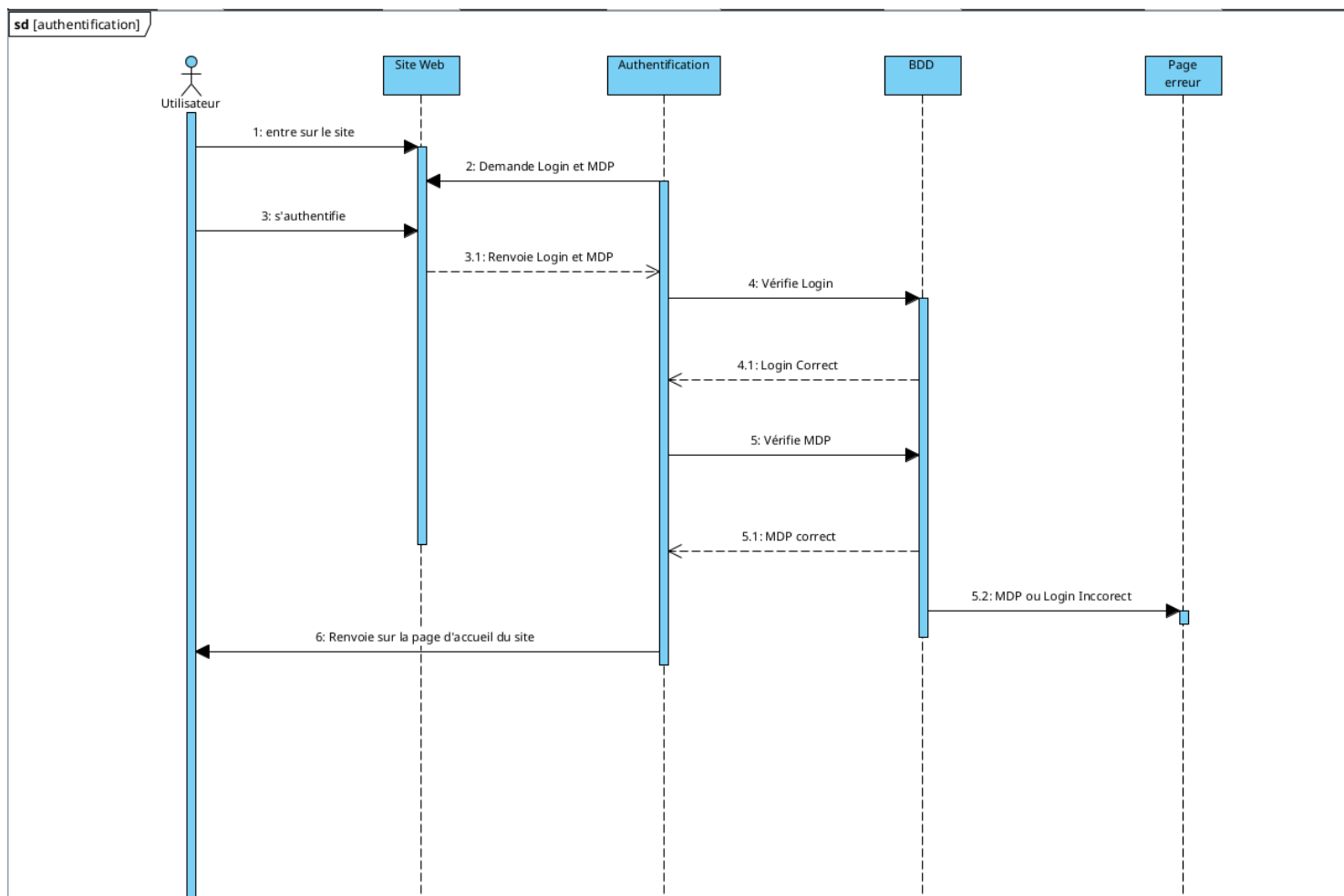
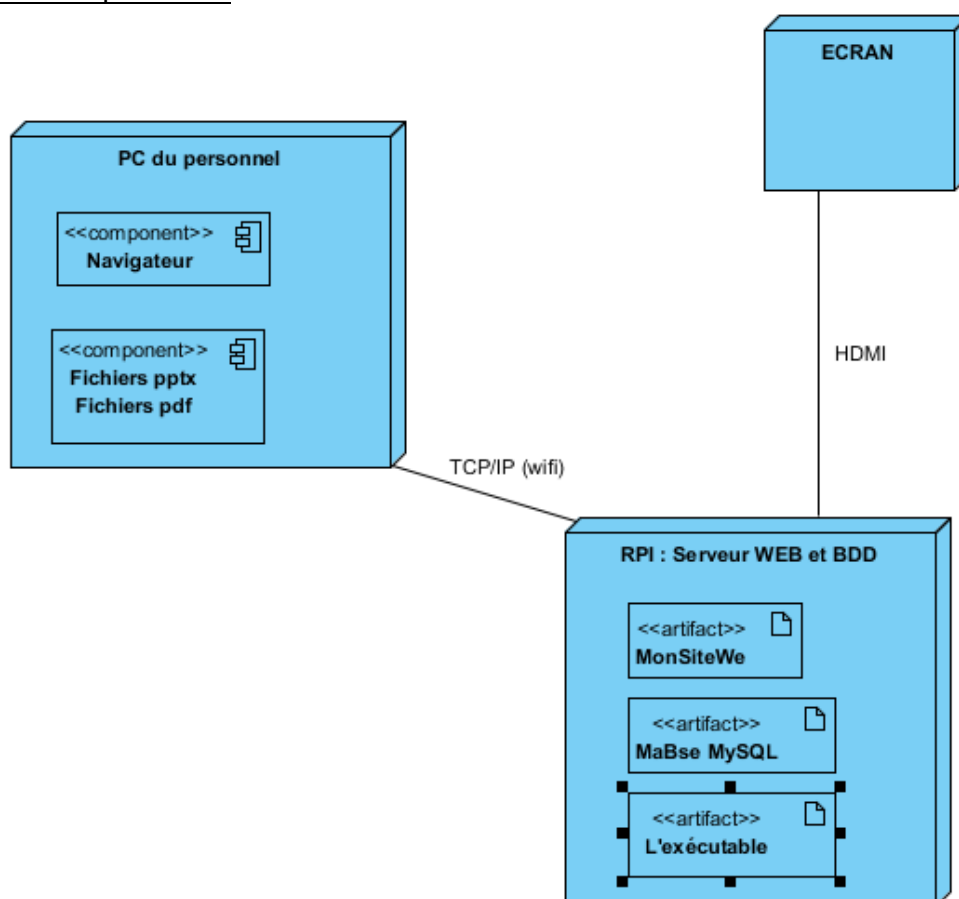
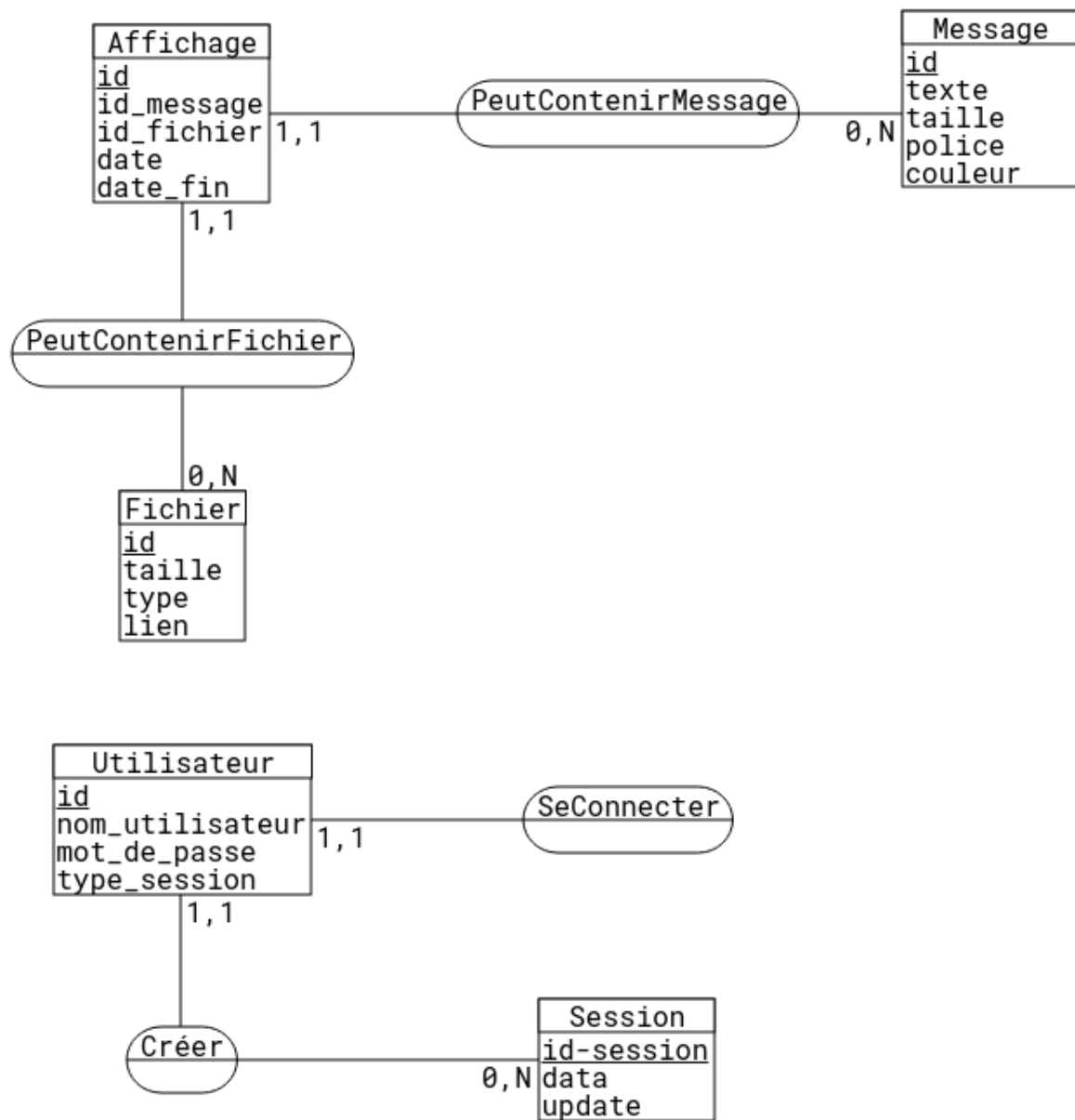


Diagramme de déploiement



Modèle Conceptuel des Données



Répartition des tâches

Diagramme Prévisionnel

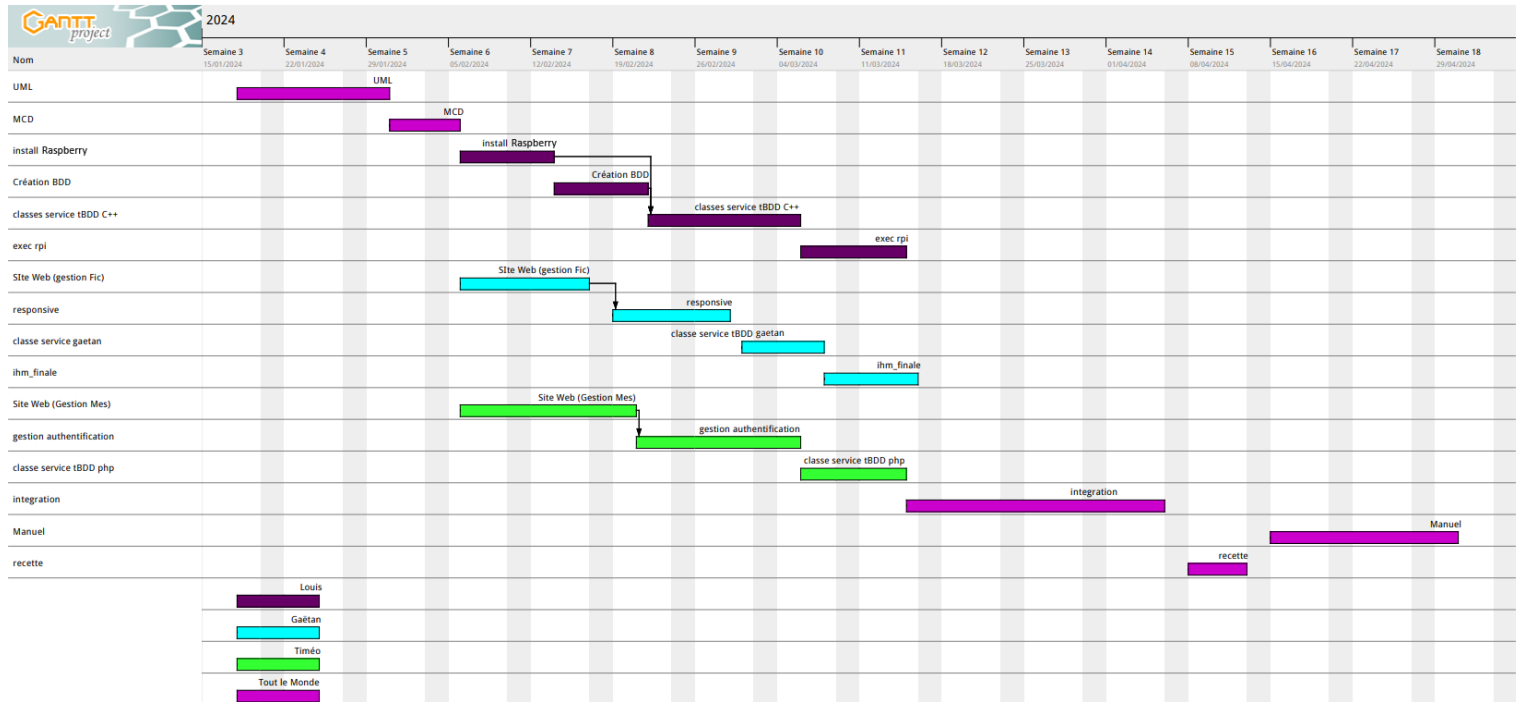
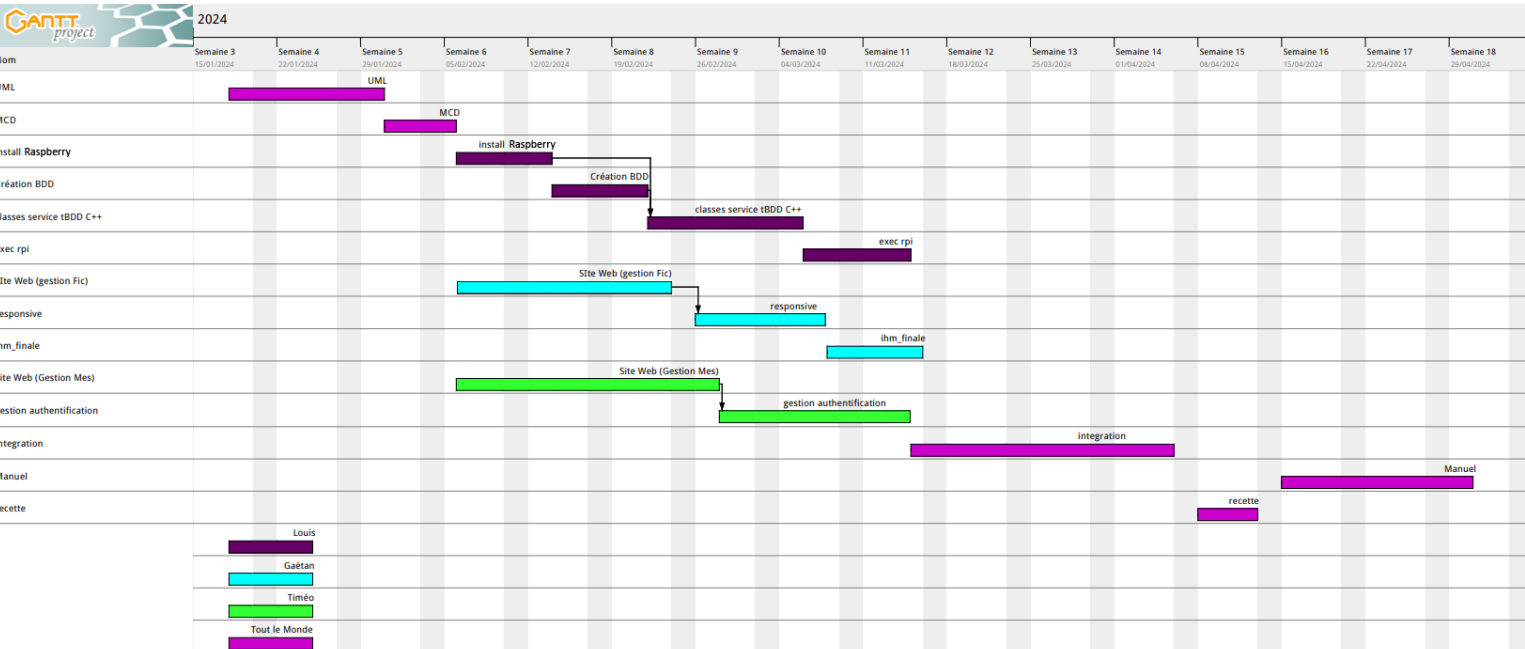


Diagramme réel



1.5- Recette globale

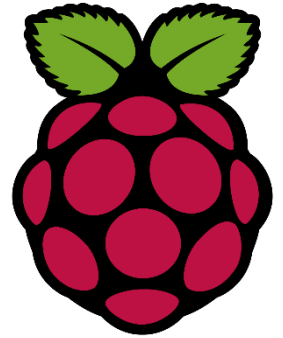
Allumer le raspberry et l'écran	Le matériel s'allume
Accéder au site internet depuis un ordinateur/smartphone	Le site renvoie sur la page de login
L'utilisateur s'authentifie	Le site renvoie sur la page d'accueil
L'utilisateur utilise le site pour envoyer un message ou un fichier et choisi une date d'affichage	Le raspberry reçoit les données
Le raspberry atteint la date pour afficher un message ou un fichier	Le message ou le fichier s'affiche sur l'écran

I- Installation du Raspberry Pi, mise en place de la base de données et réalisation de l'exécutable - Capdeville Louis

1- Installation et configuration du Raspberry Pi

Le Raspberry Pi est un nano-ordinateur possédant de nombreux avantages comme sa très petite taille, son prix abordable (moins de 100 euros selon le modèle) et sa capacité à faire ce que fait un ordinateur normal.

Nous utilisons pour ce projet un Raspberry Pi 3 qui a largement la puissance nécessaire pour notre projet d'affichage d'informations sur un écran.



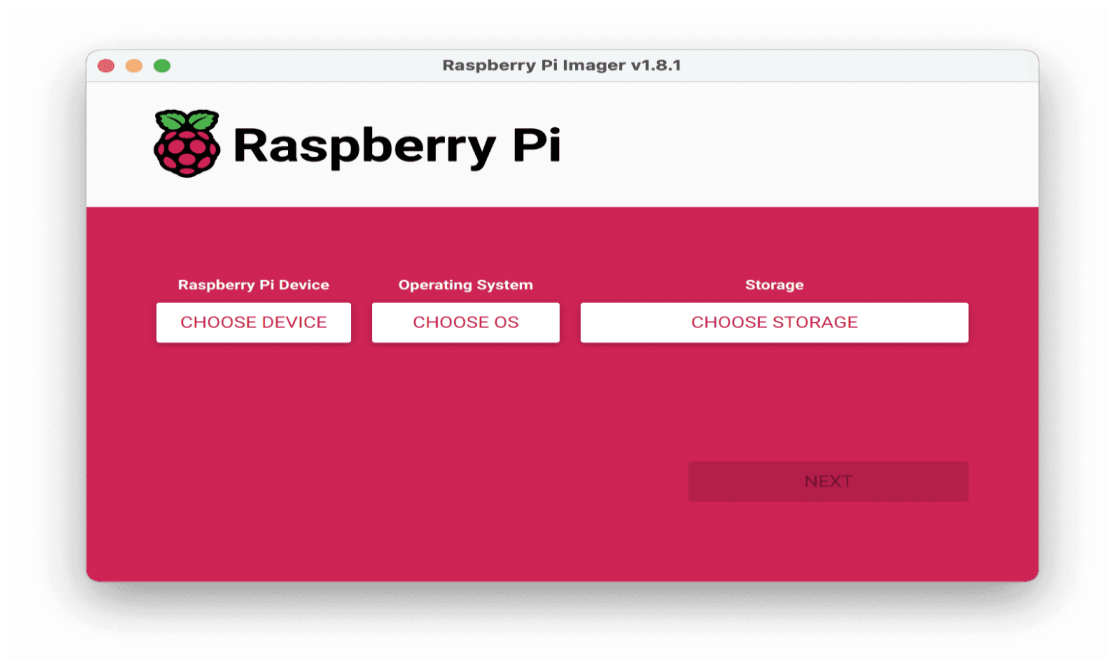
C'est sur ce Raspberry Pi que sera déployés le site internet et la base de données, et le Raspberry Pi sera branché à un écran afin de pouvoir effectuer les affichages.

1.1- Configuration de L'OS du Raspberry Pi

Pour pouvoir fonctionner notre Raspberry Pi a besoin d'un système d'exploitation ou OS (operating system) qui est un ensemble de programmes qui dirige l'utilisation des ressources d'un ordinateur par des logiciels applicatifs comme Windows ou Linux.

Raspberry Pi OS (anciennement Raspbian) est le system d'exploitation référence Raspberry Pi car c'est un OS basé sur Linux Debian (d'où Raspbian) qui a été optimisé pour les Raspberry Pis. La version de Raspberry Pi OS avec un bureau graphique qui est celle que j'ai installée dispose par défaut de Chromium qui est un navigateur internet libre qui est la base du navigateur Google Chrome, ce qui nous sera utile pour afficher nos messages et fichiers.

L'installation de Raspberry Pi OS et aussi simplifiée par l'existence de Raspberry Pi Imager, un logiciel développé par les fabricants de Raspberry Pi qui permet d'installer rapidement et facilement Raspberry Pi OS sur une carte micro-SD qui servira d'espace de stockage pour le Raspberry Pi.



1.2- Installation d'un serveur Apache2

Pour que notre Raspberry Pi puisse héberger un site web que l'on utilisera pour préparer nos affichages on a besoin d'un serveur HTTP qui permet à un site web de communiquer avec un navigateur en utilisant le protocole HTTP(S) (HyperText Transfer protocole). J'ai donc installé Apache2.



Xampp aurait pu être un moyen facile d'installer à la fois le serveur HTTP, la base de données et phpMyAdmin mais comme Xampp ne fonctionne pas sur les Raspberry Pi j'ai dû installer Apache2 en manuellement en passant par un terminal. Une fois Apache2 installé j'ai installé php, un langage de scripts libre conçu pour le développement d'applications web, que l'on utilisera sur notre site internet.

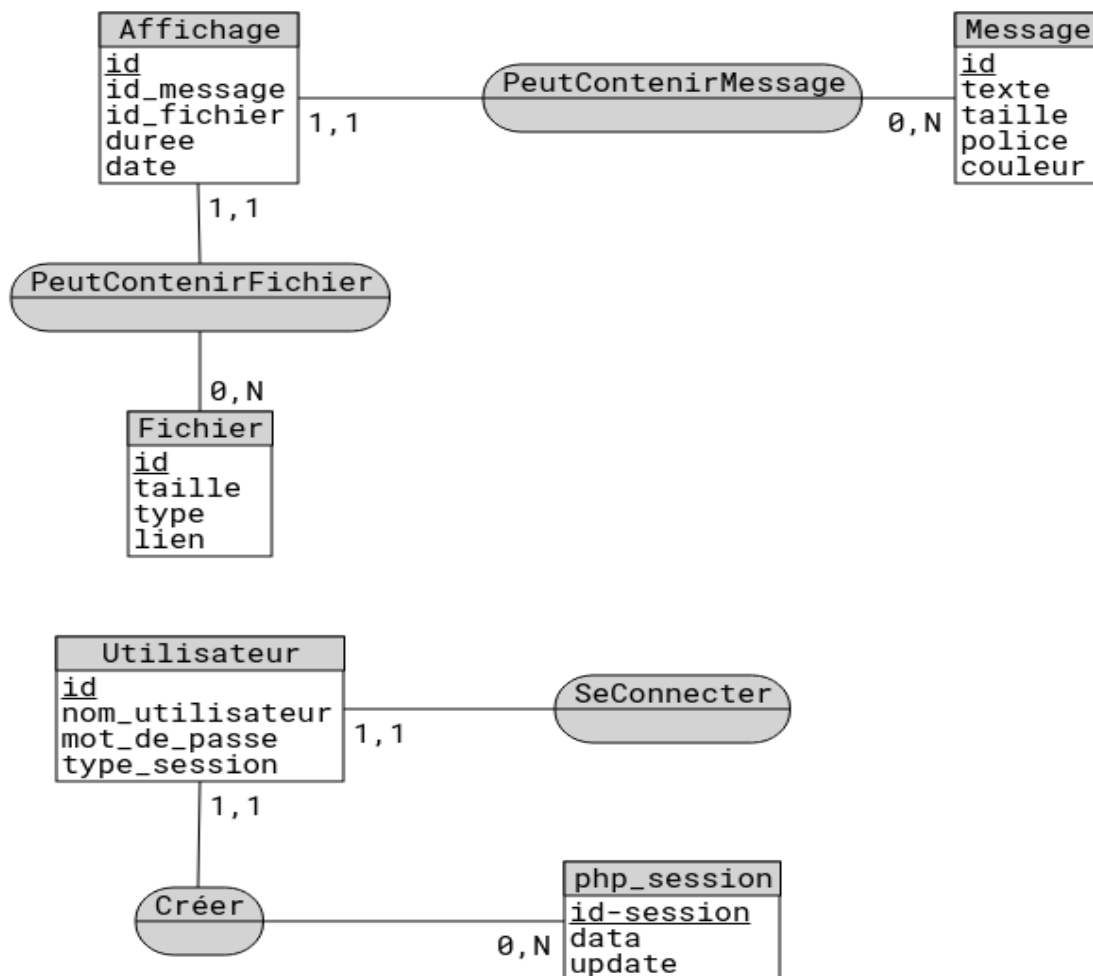
1.3- Installation de MariaDB et phpMyAdmin

Nous avons besoin d'une base de données pour notre Projet et par conséquent d'un serveur de base de données relationnelles j'ai donc installé MariaDB qui sera notre serveur et nous permettra d'utiliser du langage SQL pour interagir avec notre base de données. Puis j'ai installé phpMyAdmin, une interface simplifiée pour l'utilisation du SQL et la gestion de notre base de données.



2- Mise en place de la base de données

Après avoir installé tout le système, je me suis mis à créer la base de données selon le MCD suivant :



Sur phpMyAdmin j'ai créé la base bddProjet.

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> Affichage	Browse Structure Search Insert Empty Drop	38	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> Fichier	Browse Structure Search Insert Empty Drop	12	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> Message	Browse Structure Search Insert Empty Drop	13	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> php_session	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> utilisateur	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-

Puis j'ai fait chaque table avec les champs correspondant :

La table Affichage :

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/> 2	id_message	int(11)			Yes	NULL			Change Drop More
<input type="checkbox"/> 3	id_fichier	int(11)			Yes	NULL			Change Drop More
<input type="checkbox"/> 4	date	datetime			No	None			Change Drop More
<input type="checkbox"/> 5	date_fin	datetime			Yes	NULL			Change Drop More

La table Message :

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/> 2	texte	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 3	taille	int(11)			No	None			Change Drop More
<input type="checkbox"/> 4	police	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 5	couleur	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More

La table Fichier :

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/> 2	type	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 3	taille	int(11)			No	None			Change Drop More
<input type="checkbox"/> 4	lien	varchar(500)	utf8mb4_general_ci		No	None			Change Drop More

La table php_session :

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	id_session	int(11)			No	None			Change Drop More
<input type="checkbox"/> 2	data	varchar(1024)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 3	update	datetime			No	None			Change Drop More

La table Utilisateur :

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/> 2	nom_utilisateur	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 3	mot_de_passe	varchar(500)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 4	type_session	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More

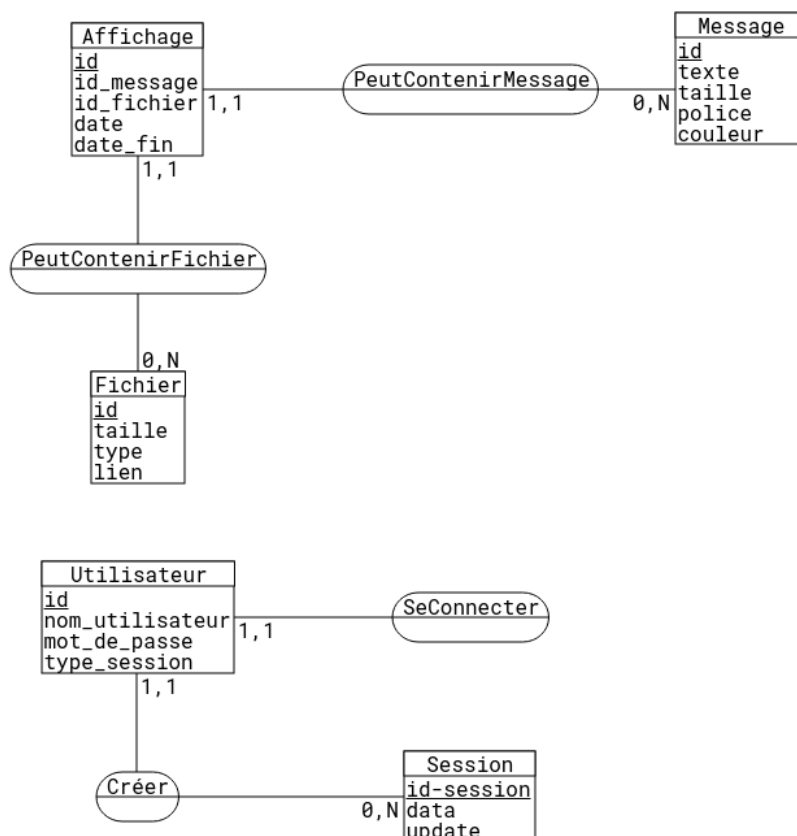
Dans la table Affichage j'ai fait en sorte que les champs idFichier et idMessage soient des clés étrangères, afin qu'ils soient liés aux ids de la table Fichier et Message respectivement, en exécutant ces requêtes SQL :

```
ALTER TABLE Affichage ADD FOREIGN KEY (idFichier) REFERENCES Fichier(id);
```

```
ALTER TABLE Affichage ADD FOREIGN KEY (idMessage) REFERENCES Message(id);
```

Ces clés étrangères seront utiles pour savoir si tel affichage prévu est un affichage de message ou de fichier.

En faisant des tests sur la base de données je me suis rendu compte que changer la table Affichage pour remplacer le champ duree par date_fin rendrait les calculs sur les dates beaucoup plus simple car date et date_fin sont tous les deux du même type qui est le date_time et la fonction NOW() du SQL rendrait les calculs plus simples, cela a donc changé le MCD pour celui-ci :



3 -Recherche de solution pour afficher

A ce moment, après avoir finis la création de la base de données j'ai cherché des solutions pour pouvoir afficher nos messages et nos fichiers sur l'écran.

J'ai cherché plusieurs logiciels (comme par exemple Zathura PDF Viewer, une visionneuse de documents gratuites) sans trouver de logiciel qui affichera comme je le voudrais, en général il y a des bordures assez marquées et le fichier ne s'affiche pas en plein écran.

Finalement je décide de choisir Chromium utilisé en mode Kiosk ce qui permet de l'ouvrir en plein écran et sans bordures. Comme Chromium est un navigateur internet on peut y ouvrir des fichiers grâce à du code html et l'on peut facilement y afficher du texte pour les messages que l'on peut styliser avec du css(Cascading Style Sheets).

Grâce à LX-Session et l'autostart je peux faire en sorte que le Raspberry Pi lance Chromium à son démarrage automatiquement et à la page internet que je veux.

Je décide d'utiliser une page cgi(Common Gateway Interface) pour permettre l'actualisation de la page d'affichage. Une page cgi est une page qui exécute un programme et retourne le contenu généré, ici le programme sera un exécutable c++ qui récupérera les données à afficher ou non et les enverra sous forme de code html pour les afficher.

Le Raspberry ouvrira donc automatiquement cette page cgi à son démarrage et cette page s'actualisera automatiquement.

4 – Programmation de la classe tBDD et création de l'exécutable

4-1 – La classe tBDD

J'ai donc programmé une classe tBDD en C++ pour récupérer les informations stockées dans la BDD(base de données).

Pour interagir avec la base de données dans du code C++ j'ai besoin de la librairie mysql.h

Initialisation des bibliothèques :

```
#include <mysql/mysql.h>
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>
#include <iostream>
#include <string>
#include <cstring>
#include <chrono>
#include <unistd.h>
```

Déclaration de la structure Affichage qui permettra de savoir quel affichage est de quel type et à quel id il renvoie :

```
struct Affichage{
    string type;

    string id;
};
```

Déclaration de la classe tBDD :

```
class tBDD
{
private:
    MYSQL* connexion;
    MYSQL_RES* resultats;
    MYSQL_ROW ligne;
    string date;
    Affichage infos;
    string idAfficher;
public:
    tBDD();
    ~tBDD();
    string testDateAffichage();
    void AfficherFichier(string idFichier);
    void testQuoiAfficher(string idAffichage, Affichage * reponse);
    void AfficherMessage(string idMessage);
    void AfficherDefault();
};
```

Constructeurs et destructeurs de la classe :

```

tBDD::tBDD()
{
    connexion = mysql_init(NULL);
    assert(connexion != NULL);
    if(mysql_real_connect(connexion, "localhost", "etudb", "etupass", "bddProjet", 0, NULL, 0) == NULL)
    {
        exit (1);
    }
}

tBDD::~tBDD()
{
    mysql_close(connexion);
}

```

Dans le constructeur on initialise la connexion à la BDD « bddProjet » et on s'assure de sa réussite puis lorsque l'on a fini d'utiliser la classe le destructeur vient fermer la connexion à la BDD.

La fonction testDateAffichage() envoie une requête SQL à la BDD qui demande à recevoir toutes les informations des affichages qui ont une date inférieure à celle de maintenant, le NOW() du SQL , et une date_fin supérieure à celle de maintenant donc la situation dans laquelle nous sommes dans l'intervalle de temps entre le début et la fin de l'affichage donc il doit s'afficher. Si nous sommes dans un intervalle où il doit y avoir un affichage la fonction renvoie l'id de l'affichage en question sinon elle renvoie 0.

TestDateAffichage() :

```

string tBDD::testDateAffichage()
{
    string id_afficher;
    if (mysql_query(connexion, "SELECT * FROM `Affichage` WHERE date < NOW() AND NOW() < date_fin;") != 0)
    {
        exit(1);
    }
    resultats = mysql_store_result(connexion);
    if (resultats == NULL)
    {
    }
    else if (mysql_num_rows(resultats) != 0)
    {
        ligne = mysql_fetch_row(resultats);
        id_afficher = ligne[0];
        mysql_free_result(resultats);
        return id_afficher;
    }
    id_afficher = "0";
    return id_afficher;
}

```

La fonction testQuoiAfficher() prend en paramètres un string qui contient un id , celui qui a été renvoyé par la fonction testDateAffichage() si elle renvoie autre chose que 0, et un pointeur d'une structure Affichage.

Elle envoie une requête SQL à la BDD qui récupère toutes les informations de la ligne qui a pour id celui pris en paramètre puis en fonction de quelle clé étrangère entre idFichier et id-Message renvoie une valeur différente de NULL elle stocke dans la structure Affichage pointée le type d'affichage entre Message et Fichier et l'id du message ou du fichier à afficher.

testQuoiAfficher() :

```
void tBDD::testQuoiAfficher(string idAffichage, Affichage* reponse)
{
    string requete = "SELECT * FROM `Affichage` WHERE id =" + idAffichage + ";";
    char* requete = new char [requete.length()+1];
    strcpy(requete, requete.c_str()); ⚠ Call to function 'strcpy' is insecure as i
    if (mysql_query(connexion, requete) != 0)
    {
        exit(1);
    }
    resultats = mysql_store_result(connexion);
    if (resultats == NULL)
    {
        exit(1);
    }
    else if (mysql_num_rows(resultats) != 0)
    {
        ligne = mysql_fetch_row(resultats);
        if (ligne[1] == NULL)
        {
            reponse->type = "Fichier";
            reponse->id=ligne[2];
        }
        else
        {
            reponse->type = "Message";
            reponse->id=ligne[1];
        }
    }
}
```

La fonction AfficherFichier() prend en paramètre un string contenant l'id du fichier à afficher. Elle envoie une requête SQL à la BDD qui demande le lien du fichier à afficher puis envoie du code html qui fait afficher le fichier grâce à la balise <embed> qui est une balise html très pratique car elle permet d'afficher des fichiers de types jpg, png, pdf, et même des mp4. Ce code html sera ensuite exécuté par la page cgi qui affichera le fichier.

AfficherFichier() :

```

void tBDD::AfficherFichier(string idFichier)
{
    string requete = "SELECT lien FROM `Fichier` WHERE id =" + idFichier + ";";
    char* requete = new char [requete.length()+1];
    strcpy(requete, requete.c_str()); Δ Call to function 'strcpy' is insecure as
    if (mysql_query(connexion, requete) != 0)
    {
        exit(1);
    }
    resultats = mysql_store_result(connexion);
    if (resultats == NULL)
    {
        exit(1);
    }
    else
    {
        ligne = mysql_fetch_row(resultats);
        printf("<body style=\"background-color:Black;\">");
        printf("<embed src=\"../FichierDL/%s\" width=\"100%%\" height=\"100%%\" al

        mysql_free_result(resultats);
    }
}

```

La fonction AfficherMessage() a un fonctionnement sensiblement similaire à la différence qu'elle demande plus d'informations dans la requête SQL et qu'elle affiche le texte en prenant en compte les informations de style qui ont été stockées dans la BDD.

AfficherMessage() :

```

void tBDD::AfficherMessage(string idMessage)
{
    string requete = "SELECT texte, taille, police,couleur FROM Message WHERE id =" + idMessage + ";";
    char* requete = new char [requete.length()+1];
    strcpy(requete, requete.c_str()); Δ Call to function 'strcpy' is insecure as it does not provide bounding of the m
    if (mysql_query(connexion, requete) != 0)
    {
        exit(1);
    }
    resultats = mysql_store_result(connexion); Δ Potential leak of memory point
    if (resultats == NULL)
    {
        exit(1);
    }
    else
    {
        ligne = mysql_fetch_row(resultats);
        printf("<body style=\"font-family:'%s';font-size:%s;color:%s;\">", ligne[2], ligne[1], ligne[3]);
        printf("<p align=\"center\" style=\"line-height: 950px;height:950px;\">%s</p> ", ligne[0]);

        mysql_free_result(resultats);
    }
}

```

La fonction AfficherDefaut() quand à elle ne prend pas de paramètre car elle sera effectuées que lorsqu'il n'y a pas d'affichage prévu maintenant.

Elle envoie une requête SQL à la BDD afin d'obtenir l'heure du système et affiche un message disant « Bonjour » et l'heure qu'il est.

AfficherDefault() :

```
void tBDD::AfficherDefault()
{
    string requete = "SELECT HOUR(now()), MINUTE(NOW()) ";
    char* requete = new char [requete.length()+1];
    strcpy(requete, requete.c_str()); // Call to function 'strcpy' is insecure as it does not provide bounding of the memory buffer. Replace with strncpy
    if (mysql_query(connexion, requete) != 0)
    {
        exit(1);
    }
    resultats = mysql_store_result(connexion); // Potential leak of memory pointed to by 'requete' [clang-analyzer-leak]
    if (resultats == NULL)
    {
        exit(1);
    }
    else
    {
        ligne = mysql_fetch_row(resultats);
        printf("<center>");
        printf("<body style='\";font-family:'Comic Sans MS';font-size:100;color:#0040ff;cursor:none;\">");
        printf("<p style='\"vertical-align:middle;height:950px;line-height: 950px;\">Bonjour, il est %s:%s</p> ", ligne[0], ligne[1]);
        printf("<center>");
        mysql_free_result(resultats);
    }
}
```

4-2 – Le main et l'exécutable

Dans le main() il n'y a que les envois des autres balises html qui vont faire s'afficher correctement la page dont une balise qui va faire se rafraîchir la page automatiquement toutes les 30 secondes, celle-ci :

```
printf("<meta charset='\"utf-8\" http-equiv='\"refresh\" content='\"30\">");
```

Et le reste du main appelle les fonctions de la classe :

D'abord il y a un test pour savoir s'il y a un affichage à faire avec la fonction testDateAffichage(), puis s'il y en a un on réalise un test pour savoir de quel type est il et quel est son id avec testQuoiAfficher(), et enfin la fonction AfficherFichier() ou la fonction AfficherMessage() est appelée pour réaliser l'affichage adéquat. Et s'il n'y avait pas d'affichage à faire le main appelle directement AfficherDefault().

Le main() :

```
int main()
{
    cout << "test" << endl;
    tBDD mabdd;
    string test;
    Affichage testAffi;
    while(1)
    {
        test = mabdd.testDateAffichage();
        cout << test << endl;
        if (test != "0")
        {
            mabdd.testQuoiAfficher(test, &testAffi);
            cout << "Mon affichage est un " << testAffi.type << " et son id est : " << testAffi.id << endl;
            if( testAffi.type == "Message")
            {
                mabdd.AfficherMessage(testAffi.id);
            }
            if (testAffi.type == "Fichier")
            {
                mabdd.AfficherFichier(testAffi.id);
            }
        }
        else
        {
            cout << "Pas d'affichage prévu.\n";
            mabdd.AfficherDefaut();
        }
        sleep(10);
    }
    return 0;
}
```

Pour l'exécutable il suffit d'exécuter la commande après avoir les codes sources sur le Raspberry Pi :

```
g++ tbdd.cpp main.cpp -lmysqlclient -o mon_executable
```

5 – Les recettes et tests

5-1 Tests unitaires

On veut tester la reconnaissance du type d'affichage ou non :

On prévoit un affichage d'un fichier dans la BDD et nous sommes le 24 mai 2024 :

				id	idMessage	idFichier	date	date_fin
<input type="checkbox"/>	Éditer	Copier	Supprimer	1	1	NULL	2024-05-21 10:00:00	2024-05-22 14:35:00
<input type="checkbox"/>	Éditer	Copier	Supprimer	2	NULL	2	2024-05-23 12:00:00	2024-05-29 00:00:00

Le 24 mai 2024 est bien dans l'intervalle entre le 23 et le 29.

On fait un code de test :

```

int main()
{
    cout << "test" << endl;
    tBDD mabdd;
    string test;
    Affichage testAffi;
    while(1)
    {
        test = mabdd.testDateAffichage();
        cout << test << endl;
        if (test != "0")
        {
            mabdd.testQuoiAfficher(test, &testAffi);
            cout << "Mon affichage est un " << testAffi.type << " et son id est : " << testAffi.id << endl;
            if( testAffi.type == "Message")
            {
                mabdd.AfficherMessage(testAffi.id);
            }
            if (testAffi.type == "Fichier")
            {
                mabdd.AfficherFichier(testAffi.id);
            }
        }
        else
        {
            cout << "Pas d'affichage prévu.\n";
            mabdd.AfficherDefault();
        }
        sleep(10);
    }
    return 0;
}

```

Le Résultat du test :

```

test
2
Mon affichage est un Fichier et son id est : 2
<body style="background-color:Black;"><embed src="../../FichierDL/test2.pdf" width="100%" height="100%" allowfullscreen="true">2

```

On a bien l'id d'affichage 2 renvoie à un fichier qui a pour id 2, puis la fonction AfficherFichier() envoie le code html.

Mais si on a cette BDD :

				id	idMessage	idFichier	date	date_fin
<input type="checkbox"/>	Éditer	Copier	Supprimer	1	1	NULL	2024-05-21 10:00:00	2024-05-22 14:35:00
<input type="checkbox"/>	Éditer	Copier	Supprimer	2	NULL	2	2024-05-23 12:00:00	2024-05-23 14:00:00

On n'a pas d'affichage prévu maintenant si on est le 24 mai.

Le résultat obtenu :

```

test
0
Pas d'affichage prévu.

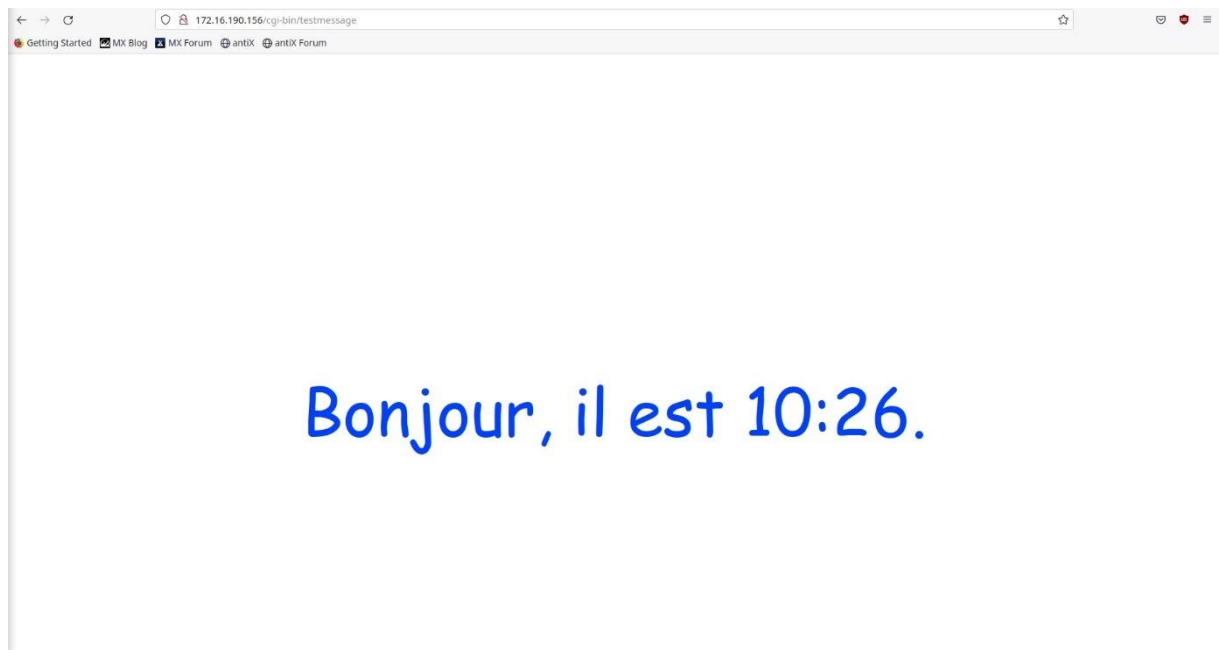
```

On a donc pas d'affichage prévu conformément à ce qu'il y a dans la BDD.

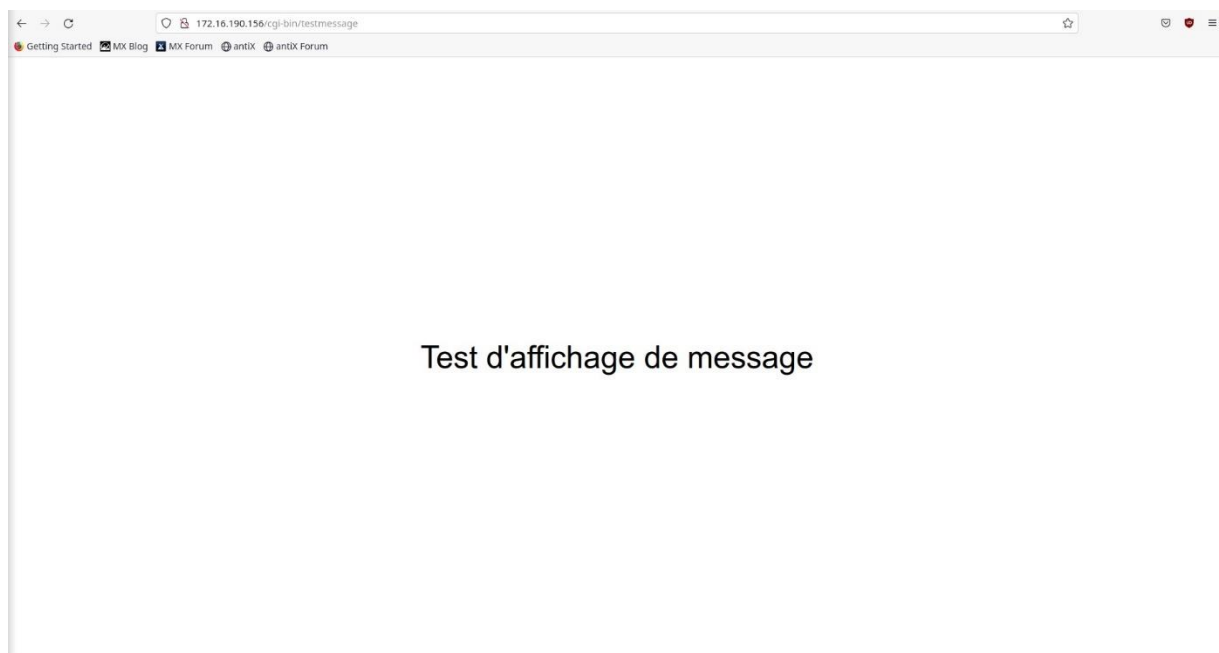
5-2 Tests d'intégrations

(Les test d'intégrations sont des captures d'écran de la page cgi ouverte sur autre ordinateur donc il y a des bords qui ne sont pas sur l'écran du Raspberry Pi)

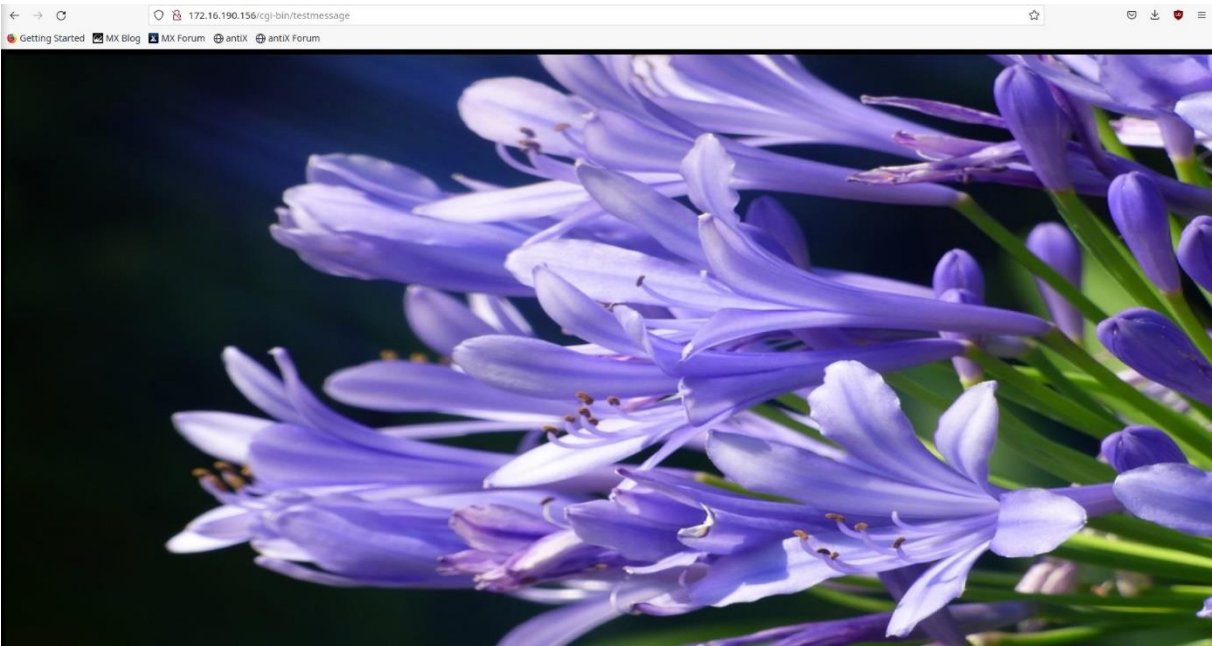
Test affichage par défaut :



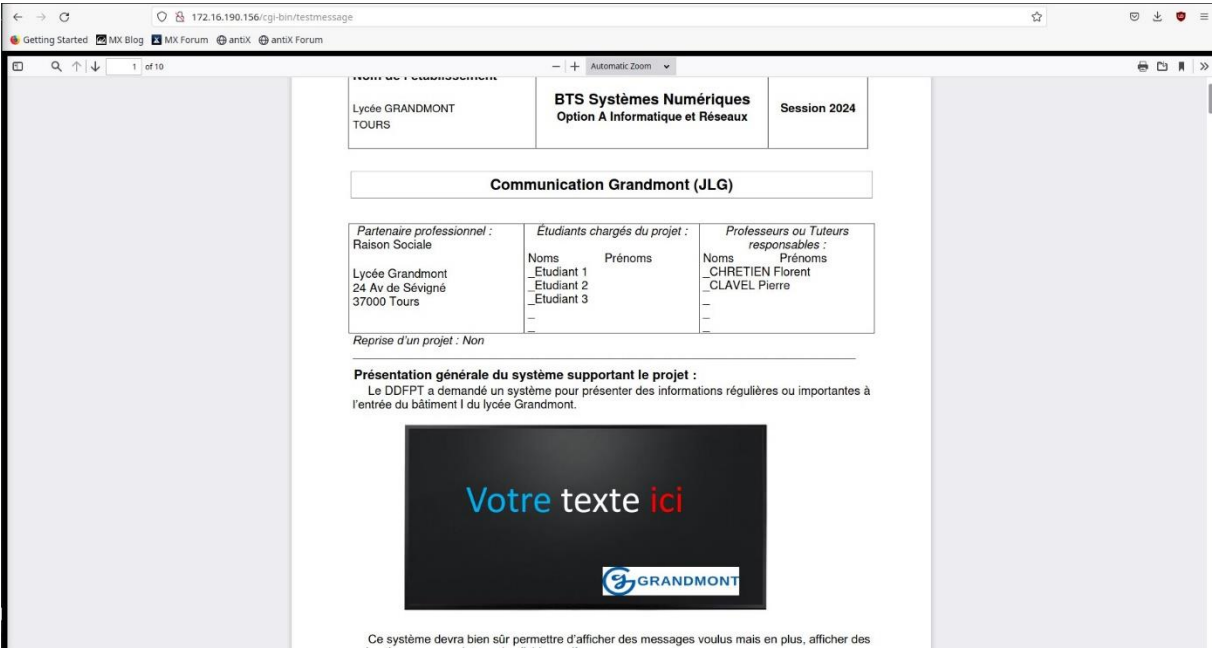
Test affichage d'un message :



Test affichage d'un fichier JPG :



Test affichage d'un fichier pdf :



5-3 Recettes

Récupérer les informations de la BDD.	Le programme récupère les informations.
Afficher la page par défaut quand il n'y a pas d'affichage prévu.	La page d'affichage par défaut s'affiche quand voulu.
Afficher un Fichier jpg/pdf/png	Les fichier jpg/pdf/png s'affichent.
Afficher un message avec les bonnes couleurs, bonne police, taille	Les messages s'affichent comme voulu.

6 – problèmes rencontrés et solutions

J'ai eu beaucoup de difficultés à choisir comment afficher les fichiers et messages et avec quel outil, j'ai finalement décidé de choisir d'utiliser une page cgi sur chromium car cela me semblait être la meilleure solution à laquelle j'ai pensé que l'on a déjà vu les pages cgi en cours ce qui m'a fait penser à notre sujet de projet.

II- Envoi de Fichiers et responsive – Gaëtan Rousseau

1- Envoi de Fichiers

Mon objectif ici consiste à transférer un fichier vers une base de données pour pouvoir l'afficher à l'écran ultérieurement. Pour ce faire, nous envisageons d'utiliser un formulaire en PHP. Cette approche est choisie car un site web est accessible sur tous les appareils sans nécessiter de déploiement sous forme d'application, ce qui le rend plus facile à utiliser et plus accessible. Une fois que le formulaire d'envoi est complété, les informations sont transmises à la base de données pour être utilisées ultérieurement dans l'affichage.

1.1- Création

La première étape est de créer le formulaire en HTML

```
<form action="envoiif.php" method="post" enctype="multipart/form-data">
<h2>Uploader un fichier</h2>
  <label for="fichier">Sélectionnez un fichier :</label>
  <input type="file" name="nomfichier" id="fichier"required>
  |   | <br>
  <input type="submit" value="Télécharger"><br>
```

Uploader un fichier

Sélectionnez un fichier :

Aucun fichier choisi

Le fichier du formulaire est téléchargé sur la machine hôte du site, et les informations du fichier sont stockées dans des variables.

```

$fic_info = pathinfo($_FILES["nomfichier"]["name"]);

$dossierTelechargement = "/var/www/html/FichierDL";
$nomFichier = basename($_FILES['nomfichier']['name']);
    $cheminFichier = $dossierTelechargement . $nomFichier;

global $extens ;
$extens = ".$fic_info['extension'];
global $taille;
$taille = filesize($fic_info['basename']);

global $newchemin;
$newchemin = $fic_info['basename'];
if ($extens == ".gif")
{
    $taille = getimagesize($nomFichier );
}

```

On se connecte ensuite à la BDD avec ce code

```

$conn = new mysqli(DB_HOST, DB_USER, DB_PSWD, DB_NAME, DB_PORT);
if ($conn->connect_error) {
    die("La connexion a échoué : " . $conn->connect_error);
}

```

DB_HOST, DB_USER, DB_PSWD, DB_NAME et DB_PORT sont des constantes contenant les informations de connexion à la base de données et sont stockées dans un autre fichier. Ici, nous créons une nouvelle instance de la classe mysqli, qui représente la connexion avec la base de données. Ensuite, nous vérifions si la connexion est réussie avec l'instruction if.

```

$req = $conn->prepare('SELECT * FROM Fichier WHERE lien = ?');

if ($req) {

    $req->bind_param('s', $fichier);

    $req->execute();

    $result = $req->get_result();
}

```

Avec cette requête préparée, nous vérifions si le fichier soumis via le formulaire existe déjà dans la base de données. Le spécificateur de type « s » dans la méthode bind_param indique que la variable liée est une chaîne de caractères (string). Si nous devons lier un entier (Integer), nous utiliserons le spécificateur de type « i ».


On va ensuite déplacer le fichier téléchargé vers un dossier pour avoir une liste des fichiers à afficher.

```
$currentLocation = '/var/www/html/'. $nomFichier;  
$newLocation = '/var/www/html/FichierDL/'. $nomFichier;  
$newLoc = '/var/www/html/FichierDL/';  
if (!file_exists($newLoc)) {  
    mkdir($newLocation, 0777, true);  
}  
$moved = rename($currentLocation, $newLocation);  
if($moved)  
{  
    echo "File moved successfully";  
}
```


Cette sélection de fichiers à afficher sera donc fait à partir d'une liste, qui est en fait un formulaire html avec un script PHP dedans.

Sélectionnez un fichier :

date de début :



date de fin :



Envoyer

Il s'agit du code pour la liste.

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  <label for="file">Sélectionnez un fichier :</label>
  <select name="file" id="file">
    <?php
      $folder = "/var/www/html/FichierDL/"; // Spécifiez le chemin vers votre dossier
      $files = scandir($folder);

      foreach ($files as $file) {
        if ($file !== '.' && $file !== '..') {
          echo "<option value=\"".$file.\">$file</option>";
        }
      }
    <?php
  </select>

```

La commande `<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">` est utilisée pour créer un formulaire HTML qui envoie des données via la méthode POST à la même page que celle où le formulaire est défini.

La variable `$folder` contient le chemin vers le fichier téléchargé précédemment.

La variable `$files` contient elle la commande `scandir` associée à la variable `$folder`.

La commande `scandir` en PHP est utilisée pour lister les fichiers et les répertoires présents dans un répertoire spécifique. Elle renvoie un tableau contenant les noms des fichiers et des répertoires.

La commande `foreach` en PHP est utilisée pour itérer sur les éléments d'un tableau (array) ou d'un objet. Elle permet de parcourir facilement chaque élément sans avoir à gérer manuellement un index ou un compteur, rendant le code plus simple et plus lisible. Pour l'affichage, on choisira une date et une heure de début d'affichage, mais on choisira aussi la date et l'heure de fin d'affichage.

```
<label for="Date-time">date de début :</label><br>
<input
  type="datetime-local"
  id="Date-time"
  name="Date-time"
  value="AAAA-MM-JJTHH:MM"
  min="2000-01-01T00:00"
  max="2100-12-31T00:00" /><br>
<label for="Date-time_fin">date de fin :</label><br>
<input
  type="datetime-local"
  id="Date-time_fin"
  name="Date-time_fin"
  value="AAAA-MM-JJTHH:MM"
  min="2000-01-01T00:00"
  max="2100-12-31T00:00" /><br>
<!--<label for="duree">Durée d'affichage du message (min)</label>
<select name="duree" id="duree">
  <option value="5">5</option>
  <option value="10">10</option>
  <option value="15">15</option>
  <option value="20">20</option>
  <option value="25">25</option>
  <option value="30">30</option>
</select>--><br>
<input type="submit" name="submit" value="Envoyer">

```

Sélectionnez un fichier :

date de début :

mai 2024

↑ ↓

							11	09
							12	10
29	30	1	2	3	4	5		
6	7	8	9	10	11	12	13	11

2- Gestion du responsive

Au vu du choix d'une site Web pour nous permettre d'y accéder de n'importe où, il fallait intégrer du responsive aux pages web pour être utilisable sur ordinateur ou sur téléphone.

2.1- Conception

Pour mettre en place le responsive j'ai fait en sorte que les boutons et le formulaire soit toujours aligner verticalement au centre et qu'il soit à 25 % de la gauche et 25 % de la droite.

```
.button1{
  background-color: #0059ff;
  color: white;
  padding: 14px 20px;
  margin: auto;
  border: medium none;
  cursor: pointer;
  width: 50%;
  text-align: center;
  position: sticky;
  left: 25%;
  right: 25%;
}
```

Création d'une classe .css de boutons pour avoir une uniformisation des boutons pour qu'il suivent tous le même responsive car créer une classe de boutons en CSS améliore la consistance visuelle, simplifie la maintenance, permet la réutilisabilité, facilite l'extension et la personnalisation, et optimise l'utilisation des préprocesseurs CSS.

```
.button{
  background-color: #0059ff;
  color: white;
  padding: 14px 20px;
  margin: 8px 0px;
  border: medium none;
  cursor: pointer;
  width: 100%;
}
```

À partir de cette classe je développe les boutons de manière plus personnelle.

```
.button1{
  background-color: #0059ff;
  color: white;
  padding: 14px 20px;
  margin: auto;
  border: medium none;
  cursor: pointer;
  width: 50%;
  text-align: center;
  position: sticky;
  left: 25%;
  right: 25%;
}
```

```
.button2, .button3{
  background-color: white ;
  color: #0059ff;
  padding: 14px 20px;
  margin: auto;
  border: medium none;
  cursor: pointer;
  width: 50%;
  text-align: center;
  position: sticky;
  left: 25%;
  right: 25%;
}
```

Le bouton 1 est séparé des boutons 2 et 3 car ses couleurs seront différentes.

Utilisation de Bootstrap qui est Framework CSS populaire qui permet de créer des interfaces web réactives et modernes de manière rapide et efficace. Voici un exemple complet d'utilisation de Bootstrap pour créer une page web simple avec une navigation, une section de contenu, et un pied de page.

Inclusion de Bootstrap :

```
<head>
  <link rel="stylesheet" href="menu.css" >
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-C6RzsynM9KDrWMeT87bh95OGHyZPhcTIXj1lw7RuBCsyn/o0j1pcV8Qyq46cDfL" crossorigin="anonymous"></script>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

Création d'un pied de page :

```
<footer class="bg-light text-center text-lg-start mt-5">
  <div class="container p-4">
    <p class="text-center">Projets Communications </p>
  </div>
</footer>
```

Le pied de page est créé à base de balise Bootstrap.

3- Cahier de recette

3.1- Fichiers

Action	Résultat attendu
Allumer le raspberry	Le matériel s'allume
Accéder au site depuis un ordinateur	Le site s'ouvre sur la page de login
Aller dans la section des fichiers	Le site renvoie sur la page des fichiers
L'utilisateur upload un fichier	Le fichier est téléchargé dans un dossier réservé sur la raspberry, le fichier est envoyé sur la base de données
Aller dans la section Liste	Le site renvoie sur la page de la liste
L'utilisateur choisit un fichier précédemment téléchargé, sélectionne une date et une heure début et de fin	Le fichier s'affiche sur l'écran à la date prévue

III- Envoi de messages et authentification – Timéo Le Bronec

1- Envoi de messages

Mon objectif ici sera de pouvoir envoyer un message dans une base de données afin de pouvoir l'afficher sur l'écran à un moment et une mise en forme choisis. Pour ce faire on envisage une solution avec un formulaire en PHP, car un site internet est disponible sur tous les appareils sans avoir besoin de le déployer partout, ce qui le rend plus simple d'utilisation et plus accessible.

Une fois le formulaire d'envoi rempli, on envoie les informations dans la base de données qui seront utilisées ultérieurement pour l'affichage.

1.1- Conception

La première étape est de créer le formulaire en HTML.



Les informations du formulaire sont renvoyées avec la méthode post, on les récupère avec ce code pour les stocker dans des variables.

```
if (isset($_POST['message']))
{
    $message = $_POST['message'];
}
else {
    echo "Le champ est obligatoire.";
}

if (isset($_POST['Date-time']))
{
    $date = $_POST['Date-time'];
}
else {
    echo "Le champ est obligatoire.";
}

if (isset($_POST['Police']))
{
    $font = $_POST['Police'];
}

if (isset($_POST['text_color']))
{
    $color = $_POST['text_color'];
}

if (isset($_POST['taille']))
{
    $taille = $_POST['taille'];
}

if (isset($_POST['Date-time_fin']))
{
    $datefin = $_POST['Date-time_fin'];
}
```


On se connecte ensuite à la BDD avec ce code.

```
$conn = new mysqli(DB_HOST, DB_USER, DB_PSWD, DB_NAME, DB_PORT);
if( $conn->connect_error ) {
    die("Erreur : 1conn->connect_error");
}
```

DB_HOST, DB_USER, DB_PSWD, DB_NAME et DB_PORT sont des constantes contenant les informations de connexion avec la BDD et sont stockées dans un autre fichier. Ici, on crée une nouvelle instance de la classe mysqli qui représente la connexion avec la base de données. On vérifie ensuite si la connexion est réussie avec le « if ».

```
$req = $conn->prepare('SELECT * FROM Message WHERE texte = ?');
if ($req) {

    $req->bind_param('s', $message);

    $req->execute();

    $result = $req->get_result();

}
```

Avec cette requête, on vérifie si le message entré dans le formulaire existe déjà dans la base de données. Le « s » dans la commande « bind_param » signifie que la variable est un « string », une chaîne de caractère, si on entre un entier, un « int », on utilise la lettre « i ».

```
$req = $conn->prepare('SELECT * FROM Affichage WHERE date = ?');
if($req){
    $req->bind_param('s', $date);

    $req->execute();

    $result_date = $req->get_result();
}
```

Avec cette requête, on vérifie si la date choisie est déjà utilisée pour un autre message, pour éviter d'avoir deux messages en même temps.

```
if (isset($_POST['message'])) {
if($result_date->num_rows > 0){
    header('Location: envoi_pas_ok.php');
    die;
}
```

Si la date choisie est déjà prise, on renvoie vers une page d'erreur avec ce code.

```

}else
if($result->num_rows > 0){
    $req= $conn->prepare('UPDATE Message SET taille = ?, police = ?, couleur = ? WHERE texte = ?');
    $req->bind_param('iss', $taille, $font, $color, $message);
    $req->execute();

    $req = $conn->prepare('SELECT id FROM Message WHERE texte = ?');
    if($req){
        $req->bind_param('s', $message);
        $req->execute();
        $result = $req->get_result();
        $row = $result->fetch_assoc();
        $id_message = $row['id'];
        $result->close();
    }
    $req = $conn->prepare('INSERT INTO Affichage SET date = ?, id_message = ?, date_fin = ?');
    $req->bind_param('sis', $date, $id_message, $datefin);
    $req->execute();
}

```

Ce code va commencer par utiliser la requête permettant de savoir si un message existe déjà avec « if(\$result->num_rows > 0) », donc si la requête a renvoyé plus de 0 résultats, c'est que le message entré existe déjà, à ce moment-là, on ne va pas créer un nouveau message dans la base de données mais simplement modifier ses informations dans la table Message et créer une nouvelle instance de l'affichage dans la table Affichage.

```

else{
    //insertion des données

    $req = $conn->prepare('INSERT INTO Message(texte, taille, police, couleur) VALUES (?, ?, ?, ?)');
    $req->bind_param('siss', $message, $taille, $font, $color);
    $req->execute();

    $id_message = $req->insert_id;

    // Préparation de la requête pour insérer dans la table Affichage
    $req = $conn->prepare('INSERT INTO Affichage(date, id_message, date_fin) VALUES(?, ?, ?)');
    $req->bind_param('sis', $date, $id_message, $datefin);
    $req->execute();
}

```

Si le message n'existe pas encore, on va cette fois-ci le créer avec cette requête.

```

header('Location: envoi_ok.php');
die;

```


Si aucune erreur n'est survenue, on renvoie l'utilisateur sur une page l'informant que l'envoi du message s'est bien passé puis on ferme la connexion à la BDD avec « die ».

2- Authentication

Afin de sécuriser ce site, il fallait installer un système d'authentification, pour que seules les personnes ayant connaissance du login et du mot de passe puisse y accéder.

2.1- Conception

Comme pour l'envoi de messages, il faut créer un formulaire HTML.



The image shows a login form with a white background and a subtle shadow. At the top, the title 'Connexion' is centered in a bold, black font. Below the title, there are two labels: 'Nom d'utilisateur' and 'Mot de passe', both in bold black text. Under 'Nom d'utilisateur' is a text input field with the placeholder text 'Entrez le nom d'utilisateur'. Under 'Mot de passe' is a text input field with the placeholder text 'Entrez le mot de passe'. At the bottom of the form is a solid blue button with the text 'Connexion' in white, centered.

Ce formulaire permet à l'utilisateur de rentrer son login et son mot de passe pour se connecter au site, les données rentrées sont comparées avec les logins et mots de passe enregistrés dans la base de données.

```

if(isset($_POST['username']) && isset($_POST['password']))
{
    // connexion à la base de données
    $db = mysqli_connect(DB_HOST, DB_USER, DB_PSWD,DB_NAME)
    or die('could not connect to database');

```

Pour vérifier si les logins sont corrects, il faut commencer par se connecter à la base de données, la condition pour commencer cette vérification est d'être sûr que les informations ont été saisies grâce à « `isset($_POST['username'])` » et « `isset($_POST['password'])` », ensuite, si cette condition est respectée, on se connecte à la BDD en crée une nouvelle instance de la classe `mysqli` avec en paramètre les constantes de connexion avec la base et coupe la connexion si elle echoue.

```

$username = mysqli_real_escape_string($db,htmlspecialchars($_POST['username']));
$password = mysqli_real_escape_string($db,htmlspecialchars($_POST['password']));

```

Ensuite, on déclare les variables `username` et `password` et on leur associe les valeurs envoyées par le formulaire avec les fonctions « `mysqli_real_escape_string` » et « `htmlspecialchars` » qui permettent d'éviter les injections SQL.

```

if($username != "" && $password != "")
{
    $requete = "SELECT count(*) FROM utilisateur where
    nom_utilisateur = '". $username.'" and mot_de_passe = '". $password.'" ";
    $exec_requete = mysqli_query($db,$requete);
    $reponse = mysqli_fetch_array($exec_requete);
    $count = $reponse['count(*)'];

```

Ensuite, si les variables `username` et `password` ne sont pas vide, on prépare une requête pour sélectionner dans la base de données le nom d'utilisateur et le mot de passe rentré, les résultats sont stockés dans la variable `count`.

```

if($count!=0) // nom d'utilisateur et mot de passe corrects
{
    $_SESSION['username'] = $username;
    header('Location: /index.php');
}
else
{
    header('Location: login.php?erreur=1'); // utilisateur ou mot de passe incorrect
}
else
{
    header('Location: login.php?erreur=2'); // utilisateur ou mot de passe vide
}
}
else
{
    header('Location: login.php');
}
mysqli_close($db); // fermer la connexion

```

La variable count est utilisée ensuite pour savoir si le nom d'utilisateur et le mot de passe sont corrects, s'il est égal à 0, c'est qu'ils sont corrects, sinon c'est qu'ils ne le sont pas. S'ils sont corrects on renvoi l'utilisateur vers la page d'accueil du site, sinon, si l'un des deux est incorrect ou manquant, le site affiche une erreur. Une fois la connexion effectuée, on ferme la connexion à la BDD.

Après qu'un utilisateur se soit authentifié, il est nécessaire de gérer sa session, sur un site web, les sessions sont stockées dans un cookie appelé « token de session » les données qu'il contient sont : la date et l'heure de la création de la session, la durée de cette session ainsi que les spécificités de l'utilisateur de la session (username, password...). Ces données sont utiles lorsqu'on veut, par exemple, limiter l'accès d'une page à un type d'utilisateur, ou, dans notre cas faire savoir au site si l'utilisateur du site s'est connecté en vérifiant s'il possède un nom d'utilisateur dans son cookie ainsi, s'il n'y a aucun username stocké, cela signifie qu'il ne s'est pas connecté, donc qu'il doit être redirigé sur la page de login.

Le code permettant de gérer une session est le suivant

```
class Session {  
  
    // Variable interne contenant la BDD  
    private $_Connexion_BDD;  
    // Initialisation de la session lors de l'appel de la classe  
    public function __construct(){  
        // Ouverture de la connexion à la BDD et association de cette  
        connexion à la variable $_Connexion_BDD  
        $this->_Connexion_BDD = connexion_bdd();  
        // Paramétrage des sessions  
        session_set_save_handler(  
            array($this, "session_ouverture"),  
            array($this, "session_fermeture"),  
            array($this, "session_lecture"),  
            array($this, "session_ecriture"),  
            array($this, "session_destruction"),  
            array($this, "session_nettoyage")  
        );  
        // Démarrage des sessions  
        session_start();  
    }  
}
```

La classe « Session » commence par se connecter à la base de données. Dans le constructeur on va paramétrer le gestionnaire de sessions grâce à « session_set_save_handler » où chaque fonction va gérer ce qui lui est associé (l'ouverture pour « session_ouverture » etc).

Session_ouverture :

```
public function session_ouverture($savePath, $sessionID) {  
    if ( $this->_Connexion_BDD ) {
```

```

        return true;
    }
    return false;
}

```

Cette fonction vérifie si la connexion avec la base de données est établie, si oui, elle renvoie « true » sinon elle renvoie « false ».

Session_Fermeture :

```

public function session_fermeture() {

    $this->session_nettoyage(ini_get("session.gc_maxlifetime"));
    $this->_Connexion_BDD = null;
    return true;
}

```

Cette fonction va nettoyer la base de données grâce à la fonction « session_nettoyage » et annule la connexion à la base en donnant la valeur « null » à « _Connexion_BDD ».

Session_lecture :

```

public function session_lecture($sessionID) {

    $datetime_actuel = new DateTime("now", new
DateTimeZone('Europe/Paris'));

    $requete = $this->_Connexion_BDD->prepare("SELECT `data` FROM
php_session WHERE `id_session` = ? LIMIT 1");

    $requete->execute([$sessionID]);

    $resultat = $requete->fetch(PDO::FETCH_ASSOC);
    if ( $resultat == true ) {
        $requete = $this->_Connexion_BDD->prepare("UPDATE php_session SET
`update` = ? WHERE `id_session` = ?");

        $requete->execute([( $datetime_actuel->format('Y-m-d H:i:s')),
$sessionID]);
        return $resultat['data'];
    };
    return '';
}

```

Cette fonction va lire les données de session stockées dans la BDD, en mettant à jour la date de modification et renvoie les données de session associées à l'ID de session, et retourne une chaîne vide si aucune donnée n'est trouvée.

Session_écriture :

```
public function session_écriture($sessionID, $sessionData) {

    $datetime_actuel = new DateTime("now", new
DateTimeZone('Europe/Paris'));

    $requete = $this->_Connexion_BDD->prepare("INSERT INTO php_session
(`id_session`, `data`, `update`) VALUES (?, ?, ?) ON DUPLICATE KEY UPDATE
`data` = ?");

    $requete->execute([$sessionID, $sessionData, ($datetime_actuel-
>format('Y-m-d H:i:s')), $sessionData]);

    $resultat = $requete->rowCount();
    if ( $resultat >= 0 ) {

        return true;
    };

    return false;
}
```

Cette fonction va gérer l'écriture des données de session dans la base de données et, en fonction de l'existence des données de liées à l'id de session, va effectuer un INSERT ou un UPDATE. Elle renvoie true ou false en fonction de la réussite ou non de l'opération.

```
public function session_destruction($sessionID) {

    $requete = $this->_Connexion_BDD->prepare("DELETE FROM php_session
WHERE `id_session` = ?");
    $requete->execute([$sessionID]);
    $resultat = $requete->rowCount();

    if ( $resultat >= 1 ) {
        return true;
    };
    return false;
}
```

La fonction gère la destruction d'une session en supprimant ses données associées de la base de données. Si la suppression réussie, on renvoie true, si quelque chose ne fonctionne pas, on renvoie false.

Session_nettoyage :

```
public function session_nettoyage($sessionMaxLifetime) {

    // Calcul du timestamp d'expiration.

    $timestamp_expiration = time() - $sessionMaxLifetime;

    // Cacul de la date d'expiration UTC.

    $date_expiration = new DateTime("@".$timestamp_expiration);

    // Formatage de la date dans le bon fuseau horaire

    $date_expiration->setTimezone(new DateTimeZone('Europe/Paris'));

    // Préparation de la requête

    $requete = $this->_Connexion_BDD->prepare("DELETE FROM php_session
WHERE `update` <= ?");

    // Execution de la requête

    $requete->execute([$date_expiration->format('Y-m-d H:i:s')]);

    // Récupération des résultats

    $resultat = $requete->rowCount();

    if ( $resultat >= 0 ) {

        // Si la suppression a réussi, on renvoie "true".

        return true;
    };
    // Si quelque chose ne fonctionne pas, on retourne "false".

    return false;
}
```

Cette fonction nettoie toutes les données de sessions ayant expirées de la base de données en supprimant les enregistrements dont la date de modification est supérieure à la date d'expiration, elle renvoie true lorsque la suppression est réussie et false dans le cas contraire.

3- Cahier de recette

3.1- Messages

Action	Résultat attendu
Allumer le raspberry	Le matériel s'allume
Accéder au site depuis un ordinateur	Le site s'ouvre sur la page de login
Aller dans la section des messages	Le site renvoie sur la page des messages
L'utilisateur rentre un message, une date de début et de fin, une taille de texte, une police et une couleur et l'envoi	Le message est envoyé sur la base de données et s'affiche sur l'écran à la date prévue

3.2- Authentification

Action	Résultat attendu
Allumer le raspberry	Le matériel s'allume
Accéder au site depuis un ordinateur	Le site s'ouvre sur la page de login
L'utilisateur ne rentre aucun login ou mot de passe	Le site affiche une erreur
L'utilisateur rentre le mauvais nom d'utilisateur et mot de passe	Le site affiche une erreur
L'utilisateur rentre les bons identifiants	Le site laisse l'utilisateur rentrer sur le site

Annexes-

Code Html formulaire envoi de messages

```
<html>

<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <link rel="stylesheet" href="test/css/menu.css" media="screen" type="text/css">
  <title>Projet - Message</title>
  <link rel="icon" type="image/gif" href="/test/img/icône.png">
  <?php

    include('login/session.php');
    session_start();
    require_once 'login/session.php';
    // Démarrage de La session
    $session = new Session();

    if($_SESSION['username'] == ""){
      header('Location:login/login.php');
      exit();
    }

  ?>

</head>

<body>

  <div id="menu">
    <div class="dropdown"> <button class="mainmenubtn">Menu</button>
      <div class="dropdown-child"> <a href="index.php">Accueil</a> <a href="form_message.php">Message</a>
      </div>
    </div>

    <div id="container">

      <form action="envoi.php" method="POST">
        <h1>Message</h1>

        <label><b>Message</b></label>
        <input type="text" placeholder="Entrer le message" name="message" required><br>

        <label for="Date-time">date de début :</label><br>

        <input
          type="datetime-local"
          id="Date-time"
          name="Date-time"
          value="AAAA-MM-JJTHH:MM"
          min="2000-01-01T00:00"
          max="2100-12-31T00:00" /><br>
        <br>

        <label for="Date-time_fin">date de fin :</label><br>
        <input
          type="datetime-local"
          id="Date-time"
          name="Date-time_fin"
          value="AAAA-MM-JJTHH:MM"
          min="2000-01-01T00:00"
          max="2100-12-31T00:00" /><br>
        <br>
        <br>
      </form>
    </div>
  </div>


```

```

<label for="font-select">Police:</label>

<select name="Police" id="font-select">
  <option value="Arial">Arial</option>
  <option value="Comicsansms">Comic sans ms</option>
  <option value="Lucida_fax">Lucida fax</option>
  <option value="Lucida_console">Lucida console</option>
  <option value="Calibri">Calibri</option>
  <option value="Timesnewroman">Times New Roman</option>
  <option value="Webdings">Webdings ???</option>
</select><br>

<label for="taille-select">Taille</label>
<select name="taille" id="taille-select">
  <option value="48">48</option>
  <option value="60">60</option>
  <option value="72">72</option>
  <option value="84">84</option>
  <option value="96">96</option>
  <option value="108">108</option>
  <option value="120">120</option>
  <option value="200">200</option>
</select><br>

<p>couleur du texte:</p>
<input type="color" id="text_color" name="text_color" value="#000000" /><br>

<input type="submit" id='submit' value='Envoyer' >

```

Code PHP envoi de message

```

include 'const.php';

error_reporting(E_ALL);
ini_set('display_errors', 1);

//r cup ration des donn es du formulaire

if (isset($_POST['message']))
{
    $message = $_POST['message'];
}
else {
    echo "Le champ est obligatoire.";
}

if (isset($_POST['Date-time']))
{
    $date = $_POST['Date-time'];
}
else {
    echo "Le champ est obligatoire.";
}

if (isset($_POST['Police']))
{
    $font = $_POST['Police'];
}

if (isset($_POST['text_color']))
{
    $color = $_POST['text_color'];
}

if (isset($_POST['taille']))
{
    $taille = $_POST['taille'];
}

if (isset($_POST['Date-time_fin']))
{
    $datefin = $_POST['Date-time_fin'];
}

```

```

$conn = new mysqli(DB_HOST, DB_USER, DB_PSWD, DB_NAME, DB_PORT);
if( $conn->connect_error ) {
    die("Erreur : 1conn->connect_error");
}

$req = $conn->prepare('SELECT * FROM Message WHERE texte = ?');
if ($req) {

    $req->bind_param('s', $message);

    $req->execute();

    $result = $req->get_result();

}

$req = $conn->prepare('SELECT * FROM Affichage WHERE date = ?');
if($req){
    $req->bind_param('s', $date);

    $req->execute();

    $result_date = $req->get_result();
}

```

```

if (isset($_POST['message'])) {
if($result_date->num_rows > 0){
    header('Location: envoi_pas_ok.php');
    die;
}else
if($result->num_rows > 0){
    $req= $conn->prepare('UPDATE Message SET taille = ?, police = ?, couleur = ? WHERE texte = ?');
    $req->bind_param('iss', $taille, $font, $color, $message);
    $req->execute();

    $req = $conn->prepare('SELECT id FROM Message WHERE texte = ?');
    if($req){
        $req->bind_param('s', $message);
        $req->execute();
        $result = $req->get_result();
        $row = $result->fetch_assoc();
        $id_message = $row['id'];
        $result->close();
    }
    $req = $conn->prepare('INSERT INTO Affichage SET date = ?, id_message = ?, date_fin = ?');
    $req->bind_param('sis', $date, $id_message, $datefin);
    $req->execute();
}
}

```

```

else{
//insertion des données

    $req = $conn->prepare('INSERT INTO Message(texte, taille, police, couleur) VALUES (?, ?, ?, ?)');
    $req->bind_param('siss', $message, $taille, $font, $color);
    $req->execute();

    $id_message = $req->insert_id;

    // Préparation de la requête pour insérer dans la table Affichage
    $req = $conn->prepare('INSERT INTO Affichage(date, id_message, date_fin) VALUES(?, ?, ?)');
    $req->bind_param('sis', $date, $id_message, $datefin);
    $req->execute();
}
}

```

Code HTML page login

```
<html>
<head>
  <meta charset="utf-8">
  <!-- importer le fichier de style -->
  <link rel="stylesheet" href="style.css" media="screen" type="text/css" />
</head>
<body>
  <div id="container">
    <!-- zone de connexion -->

    <form action="verification.php" method="POST">
    <form action="session.php" method="POST">
    <h1>Connexion</h1>

    <label><b>Nom d'utilisateur</b></label>
    <input type="text" placeholder="Entrer le nom d'utilisateur" name="username" required>

    <label><b>Mot de passe</b></label>
    <input type="password" placeholder="Entrer le mot de passe" name="password" required>

    <input type="submit" id='submit' value='Connexion' >

    <?php
    if(isset($_GET['erreur'])){
      $err = $_GET['erreur'];
      if($err==1 || $err==2)
        echo "<p style='color:red'>Utilisateur ou mot de passe incorrect</p>";
    }
    ?>

    </form>
  </div>
</body>
</html>
```

Code PHP Vérification

```
session_start();

require_once 'session.php';
// Démarrage de la session
$session = new Session();

if(isset($_POST['username']) && isset($_POST['password']))
{
  // connexion à la base de données
  $db = mysqli_connect(DB_HOST, DB_USER, DB_PSWD, DB_NAME)
  or die('could not connect to database');

  // on applique les deux fonctions mysqli_real_escape_string et htmlspecialchars
  // pour éliminer toute attaque de type injection SQL et XSS
  $username = mysqli_real_escape_string($db, htmlspecialchars($_POST['username']));
  $password = mysqli_real_escape_string($db, htmlspecialchars($_POST['password']));

  if($username != "" && $password != "")
  {
    $requete = "SELECT count(*) FROM utilisateur where
    nom_utilisateur = '". $username. "' and mot_de_passe = '". $password. "' ";
    $exec_requete = mysqli_query($db, $requete);
    $reponse = mysqli_fetch_array($exec_requete);
    $count = $reponse['count(*)'];
```

```

if($count!=0) // nom d'utilisateur et mot de passe corrects
{
    $_SESSION['username'] = $username;
    header('Location: /index.php');
}
else
{
    header('Location: login.php?erreur=1'); // utilisateur ou mot de passe incorrect
}
}
else
{
    header('Location: login.php?erreur=2'); // utilisateur ou mot de passe vide
}
}
else
{
    header('Location: login.php');
}
}
mysqli_close($db); // fermer la connexion
?>

```

Code PHP gestion des sessions

```

include ('/var/www/html/const.php');
function connexion_bdd() {

    $bdd_option['PDO::ATTR_EMULATE_PREPARES']='FALSE';
    $bdd_option['PDO::ATTR_ERRMODE']='PDO::ERRMODE_EXCEPTION';
    $bdd_option['PDO::ATTR_DEFAULT_FETCH_MODE']='PDO::FETCH_ASSOC';

    try {
        $Connexion_BDD = new PDO(BDD_DNS, 'etudb', 'etupass', $bdd_option);
    }
    catch (PDOException $e) {
        print "Erreur de connexion à la BDD ! Message : " . $e->getMessage() . "<br/>";
        die();
    }

    return $Connexion_BDD;
};

```

```

class Session {

    // Variable interne contenant la BDD
    private $_Connexion_BDD;
    // Initialisation de la session lors de l'appel de la classe
    public function __construct(){
        // Ouverture de la connexion à la BDD et association de cette connexion à la variable $_Connexion_BDD
        $this->_Connexion_BDD = connexion_bdd();
        // Paramétrage des sessions
        session_set_save_handler(
            array($this, "session_ouverture"),
            array($this, "session_fermeture"),
            array($this, "session_lecture"),
            array($this, "session_ecriture"),
            array($this, "session_destruction"),
            array($this, "session_nettoyage")
        );
        // Démarrage des sessions
        session_start();
    }
}

```

```
// Ouverture des sessions
public function session_ouverture($savePath, $sessionID) {

    if ( $this->_Connexion_BDD ) {

        // Si la connexion existe, on renvoie "true".
        return true;
    }

    // En cas d'erreur, on force php à annuler l'utilisation des sessions.
    return false;
}

```

```
// Fermeture des sessions
public function session_fermeture() {

    // Nettoyage de la BDD lors de la fermeture pour ne pas attendre le nettoyage automatique
    $this->session_nettoyage(ini_get("session.gc_maxlifetime"));

    // Destruction de la connexion
    $this->_Connexion_BDD = null;

    // Renvoie de "true" pour valider la fermeture.
    return true;
}

```

```
// Lecture des sessions
public function session_lecture($sessionID) {

    // Création d'un date_time actuel
    $datetime_actuel = new DateTime("now", new DateTimeZone('Europe/Paris'));

    // Préparation de la requête
    $requete = $this->_Connexion_BDD->prepare("SELECT `data` FROM php_session WHERE `id_session` = ? LIMIT 1");

    // Execution de la requête
    $requete->execute([$sessionID]);

    // Récupération des résultats
    $resultat = $requete->fetch(PDO::FETCH_ASSOC);
    if ( $resultat == true ) {

        // Mise à jour de la date
        // Préparation de la requête

        $requete = $this->_Connexion_BDD->prepare("UPDATE php_session SET `update` = ? WHERE `id_session` = ?");

        // Execution de la requête
        $requete->execute([( $datetime_actuel->format('Y-m-d H:i:s')), $sessionID]);

        return $resultat['data'];
    }
};

// Si quelque chose ne fonctionne pas, on ne retourne rien.
return '';

```

```
// Ecriture des sessions
public function session_ecriture($sessionID, $sessionData) {

    // Création d'un date_time actuel
    $datetime_actuel = new DateTime("now", new DateTimeZone('Europe/Paris'));

    // Préparation de la requête d'INSERT avec UPDATE si la données existe déjà
    $requete = $this->Connexion_BDD->prepare("INSERT INTO php_session (`id_session`, `data`, `update`) VALUES (?, ?, ?) ON DUPLICATE KEY UPDATE `data` = ?");

    // Execution de la requête
    $requete->execute([$sessionID, $sessionData, ($datetime_actuel->format('Y-m-d H:i:s')), $sessionData]);

    // Récupération des résultats
    $resultat = $requete->rowCount();
    if ( $resultat >= 0 ) {
        // Si L'INSERT ou L'UPDATE réussi, on renvoie "true".
        return true;
    };
    // Si quelque chose ne fonctionne pas, on retourne "false".
    return false;
}
```

```
// Destruction des sessions
public function session_destruction($sessionID) {

    // Préparation de la requête
    $requete = $this->Connexion_BDD->prepare("DELETE FROM php_session WHERE `id_session` = ?");

    // Execution de la requête
    $requete->execute([$sessionID]);

    // Récupération des résultats
    $resultat = $requete->rowCount();

    if ( $resultat >= 1 ) {
        // Si la suppression a réussi, on renvoie "true".
        return true;
    };

    // Si quelque chose ne fonctionne pas, on retourne "false".
    return false;
}
```

```
// Nettoyage de La BDD
public function session_nettoyage($sessionMaxLifetime) {

    // Calcul du timestamp d'expiration.
    $timestamp_expiration = time() - $sessionMaxLifetime;

    // Calcul de la date d'expiration UTC.
    $date_expiration = new DateTime("@.$timestamp_expiration");

    // Formatage de la date dans le bon fuseau horaire
    $date_expiration->setTimezone(new DateTimeZone('Europe/Paris'));

    // Préparation de la requête
    $requete = $this->Connexion_BDD->prepare("DELETE FROM php_session WHERE `update` <= ?");

    // Execution de la requête
    $requete->execute([$date_expiration->format('Y-m-d H:i:s')]);

    // Récupération des résultats
    $resultat = $requete->rowCount();

    if ( $resultat >= 0 ) {
        // Si la suppression a réussi, on renvoie "true".
        return true;
    };
    // Si quelque chose ne fonctionne pas, on retourne "false".
    return false;
}
```


Le test :

```
int main()
{
    cout << "test" << endl;
    tBDD mabdd;
    string test;
    Affichage testAffi;
    while(1)
    {
        test = mabdd.testDateAffichage();
        cout << test << endl;
        if (test != "0")
        {
            mabdd.testQuoiAfficher(test, &testAffi);
            cout << "Mon affichage est un " << testAffi.type << " et son id est : " << testAffi.id << endl;
            if( testAffi.type == "Message")
            {
                mabdd.AfficherMessage(testAffi.id);
            }
            if (testAffi.type == "Fichier")
            {
                mabdd.AfficherFichier(testAffi.id);
            }
        }
        else
        {
            cout << "Pas d'affichage prévu.\n";
            mabdd.AfficherDefault();
        }
        sleep(10);
    }
    return 0;
}
```

tbdd.h :

```
#ifndef TBDD_H
#define TBDD_H
#include <mysql/mysql.h>
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>
#include <iostream>
#include <string>
#include <cstring>
#include <chrono>
#include <unistd.h>

using namespace std;

struct Affichage{
    string type;

    string id;
};

class tBDD
{
private:
    MYSQL* connexion;
    MYSQL_RES* resultats;
    MYSQL_ROW ligne;
    string date;
    Affichage infos;
    string idAfficher;
public:
    tBDD();
    ~tBDD();
    string testDateAffichage();
    void AfficherFichier(string idFichier);
    void testQuoiAfficher(string idAffichage, Affichage * reponse);
    void AfficherMessage(string idMessage);
    void AfficherDefaut();
};

#endif // TBDD_H
```

tbdd.cpp

```

#include "tbdd.h"

tBDD::tBDD()
{
    connexion = mysql_init(NULL);
    assert(connexion != NULL);
    if(mysql_real_connect(connexion, "localhost", "etudb", "etupass", "bddProjet", 0, NULL, 0) == NULL)
    {
        exit(1);
    }
}

tBDD::~tBDD()
{
    mysql_close(connexion);
}

string tBDD::testDateAffichage()
{
    string id_afficher;
    if (mysql_query(connexion, "SELECT * FROM `Affichage` WHERE date < NOW() AND NOW() < date_fin;") != 0)
    {
        exit(1);
    }
    resultats = mysql_store_result(connexion);
    if (resultats == NULL)
    {
    }
    else if (mysql_num_rows(resultats) != 0)
    {
        ligne = mysql_fetch_row(resultats);
        id_afficher = ligne[0];
        mysql_free_result(resultats);
        return id_afficher;
    }
    id_afficher = "0";
    return id_afficher;
}

void tBDD::testQuoiAfficher(string idAffichage, Affichage* reponse)
{
    string requete = "SELECT * FROM `Affichage` WHERE id =" + idAffichage + ";";
    char* requete = new char [requete.length()+1];
    strcpy(requete, requete.c_str()); // Call to function 'strcpy' is insecure as i
    if (mysql_query(connexion, requete) != 0)
    {
        exit(1);
    }
    resultats = mysql_store_result(connexion);
    if (resultats == NULL)
    {
        exit(1);
    }
    else if (mysql_num_rows(resultats) != 0)
    {
        ligne = mysql_fetch_row(resultats);
        if (ligne[1] == NULL)
        {
            reponse->type = "Fichier";
            reponse->id=ligne[2];
        }
        else
        {
            reponse->type = "Message";
            reponse->id=ligne[1];
        }
    }
}

```

```

void tBDD::AfficherFichier(string idFichier)
{
    string requete = "SELECT lien FROM `Fichier` WHERE id =" + idFichier + ";";
    char* requete = new char [requete.length()+1];
    strcpy(requete, requete.c_str()); ⚠ Call to function 'strcpy' is insecure as
    if (mysql_query(connexion, requete) != 0)
    {
        exit(1);
    }
    resultats = mysql_store_result(connexion);
    if (resultats == NULL)
    {
        exit(1);
    }
    else
    {
        ligne = mysql_fetch_row(resultats);
        printf("<body style=\"background-color:Black;\">");
        printf("<embed src=\"../FichierDL/%s\" width=\"100%%\" height=\"100%%\" al

        mysql_free_result(resultats);
    }
}

```

```

void tBDD::AfficherMessage(string idMessage)
{
    string requete = "SELECT texte, taille, police,couleur FROM Message WHERE id =" + idMessage + ";";
    char* requete = new char [requete.length()+1];
    strcpy(requete, requete.c_str()); ⚠ Call to function 'strcpy' is insecure as it does not provide bounding of the m
    if (mysql_query(connexion, requete) != 0)
    {
        exit(1);
    }
    resultats = mysql_store_result(connexion); ⚠ Potential leak of memory point
    if (resultats == NULL)
    {
        exit(1);
    }
    else
    {
        ligne = mysql_fetch_row(resultats);
        printf("<body style=\"font-family:'%s';font-size:%s;color:%s;\">", ligne[2], ligne[1], ligne[3]);
        printf("<p align=\"center\" style=\"line-height: 950px;height:950px;\">%s</p> ", ligne[0]);

        mysql_free_result(resultats);
    }
}

```

```

void tBDD::AfficherDefault()
{
    string requete = "SELECT HOUR(now()), MINUTE(NOW()) ";
    char* requete = new char [requete.length()+1];
    strcpy(requete, requete.c_str()); ⚠ Call to function 'strcpy' is insecure as it does not provide bounding of the memory buffer. Replace unt
    if (mysql_query(connexion, requete) != 0)
    {
        exit(1);
    }
    resultats = mysql_store_result(connexion); ⚠ Potential leak of memory pointed to by 'requete' [clan
    if (resultats == NULL)
    {
        exit(1);
    }
    else
    {
        ligne = mysql_fetch_row(resultats);
        printf("<center>");
        printf("<body style=\"font-family:'Comic Sans MS';font-size:100;color:#0040ff;cursor:none;\">");
        printf("<p style=\"vertical-align:middle;height:950px;line-height: 950px;\">Bonjour, il est %s:%s</p> ", ligne[0], ligne[1]);
        printf("<center>");
        mysql_free_result(resultats);
    }
}

```

Le main :

```
#include <mysql/mysql.h>
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>
#include <iostream>
#include <chrono>
#include <unistd.h>
#include "tbdd.h"

using namespace std;

int main()
{
    tBDD mabdd;
    string test;
    Affichage testAffi;

    printf ("Content-Type: text/html\n\n");
    printf ("<html>\n" );
    printf ("<head>\n" );
    printf ("<meta charset=\"utf-8\" http-equiv=\"refresh\" content=\"30\">");
    printf ("</head>\n" );

    test = mabdd.testDateAffichage();
    if (test != "")
    {
        mabdd.testQuoiAfficher(test, &testAffi);

        if (testAffi.type == "Message")
        {
            mabdd.AfficherMessage(testAffi.id);
        }
        else if (testAffi.type == "Fichier")
        {
            mabdd.AfficherFichier(testAffi.id);
        }
    }
    else
    {
        mabdd.AfficherDefaut();
    }
    printf ("</body>\n" );
    printf ("</html>\n" );
    return 0;
}
```