Slide 3

In this period where the temperature is abnormally high and people use more and more fossil energy in their current life.

Our co2 emission are very important and impactful for the environment.

LANCE IMAGE 1

As we can live now, climatic warming is more and more visible

That's why, I choose to create a platform who will called « Change ton climat » L ANCE 2

to fight against the global warming with simple and playful environmental awareness for change all of that.

Slide 4

So I'm building a Website LANCE 1

Where visitors can inform themselves LANCE 2

About this subject and they can become actors LANCE 3

of the climatic transition thanks to tip sheets to reduice their carbon footprint.

Through Forums, they also can discuss LANCE 4

with other users to deal with questions and solutions.

A few technical contrainst: the website will be scalable for all device LANCE 5

and accessible to all the people so the font will change for the dyslexic persons and the color for the color-blind ones . LANCE 6

The app will be built with HTML CSS Javascript and Bootstrap for the front end. LANCE 7

For the Back end, I will use PHP Language.

For the webhost I will choose Infomaniak or O2switch because they are the best eco friendly web hoster whose use renewable energies

And now The rest of presentation will be in French

Slide 5

L'arborescence d'un site est une représentation graphique de sa structure organisationnelle. Elle va décrire la manière dont les différentes pages sont organisées et liées entre elle.

Durant cet entretien, nous allons surtout abordé les pages Chiffres et Messagerie.

Slide 6

Avant de commencer le développement, je suis passé par la phase de maquettage du projet.

Le maquettage est un process de conception visuelle pour créer des représentations graphiques ou interactive avant la mise en dev.

Il y a généralement 3 étapes : Le zoning, le Wireframe et le mockup

Le Zoning va permettre de définir la structure générale de l'application.

Le wireframe, comme vous pouvez le voir ici, va reprendre la structure du zoning et va implémenter la disposition des éléments d'interface sans se soucier de l'aspect esthétique. Il va architecturer la navigation, l'information, les zones fonctionnelles et les interactions basiques ce qui va donner une fondation solide pour la prochaine étape : Le Mockup

Slide 7

Le mockup sera la phase finale du maquettage. Elle va donner une représentation fidèle et précise du projet.
On y inclut les couleurs , les polices les éléments graphiques qui serviront de support pour la présentation de validation auprès du client mais aussi de support pour le développeur.

Slide 8

Ici une présentation en mise en situation du projet sur différent support.

Slide 9

Cas d'utilisation : Modélisation qui permet de décrire les interactions entre les utilisateurs (acteurs) et le Système d'information..

Slide 10

Acteur : Ce sont tous les utilisateurs externes au système, cela peut être des humain ou d'autre programmes

Héritage ou généralisation, est une relation entre un cas d'utilisation spécifique et un autre plus générale. Il permet de partager les droits entre plusieurs acteurs

Cas d'utilisation : Modélisation des fonctionnalités qu'offre un système au utilisateur.

Extend : c’est la relation entre 2 use case. Cette relation est non obligatoire pour le déroulement de l’utilisation

Slide 11

Include c’est la relation entre 2 use case. Cette relation est obligatoire pour le déroulement de l’utilisation

Slide 12

Le diagramme d’activité est un diagramme UML (Unified Modeling Language) de modélisation comportementale. Il permet de représenter graphiquement les différentes étapes liées au déroulement d’un process.

Etat initial : C’est le point de départ, il est toujours placé dans la partie système qui est dans l’attente d’une action au moment T

Accept signal : Il représente l’étape qui reçoit le signal entrant ou un évènement externe qui permet de pouvoir continuer l’exécution. Il spécifie le déclencheur du cas d’utilisation

Les actions sont toutes les différentes tâches de chaque étape du process. Il est représenté par un rectangle et une phrase indiquant l’action

Les décisions : Représenté par un losange, il est le point de décision dans le process où le flux d’activité peut diverger en fonction d’une condition ou d’une décision

Nœud de fin de flot : Représenté par une croix dans un cercle, il indique la fin d’un flux d’activité sur une branche ce qui n’a aucune incidence sur le reste des flux actifs de l’activités

Noeud de bifurcation : représente la décomposition du flux d’activité en plusieurs flux parallèles qui s’exécute en même temps.

Send signal : représente, quant à lui, une action qui déclenche l’envoi d’un signal ou d’un évènement à un autre

L’état final , indique la fin du process.

Slide 13

Le diagramme de séquence est une modélisation qui montre comment différents objet ou composant interagissent les uns les autres dans un ordre chronologique spécifiques. On le représente dans un scénario nominal, c’est-à-dire sans aucune erreur

Il est composé de plusieurs éléments :

La Lifeline : qui représente l'existence temporelle d'un objet ou d'une entité

Le message synchrone est un message où l'émetteur attend une réponse du système

Le self message est un message envoyé par un objet à lui-même, il indique que l'objet declenche une action qui n'interfère pas avec un autre objet ET le reply message qui est un message de retour envoyé par le système en réponse au message envoyé par l'émetteur

Les scénarios alternatifs représentent une séquence d'action qui se produit dans des conditions spécifiques qui différent du scénario nominal.

Les scénarios d'erreurs représentent une séquence d'action qui se produit lorsque des erreurs ou des exceptions ont lieu pendant le scénario principal et qui y mettent un terme.

PRESENTATION DU SITE

Slide 14

Maintenant , je vais vous présentez la partie de développement Front. L'application a été créé sous le format MVC Model View Controller

C'est un type d'architecture qui divise une application en 3 composants interconnectés qui permet une meilleure lisibilité du code , il facilite la gestion , la maintenance et l'évolutilité du code.

Le Model va répertorier l'ensemble des fichiers qui servent à représenter la structure des données et la logique métier de l'application, Les model interagissent avec la BDD.

Le View va lui centraliser les fichiers responsables de l'interface utilisateur et de l'affichage des données

Le Controller lui va inventorier les fichiers responsables des requêtes utilisateur et de la coordination entre les model et les vues

Voyons maintenant la fonction de compteur.

Slide 15

Pour commencer, je vais gérer le Template qui sert de structure à chaque vue en instanciant l'ensemble des éléments nécessaire au head.

Les Meta charset est utilisé pour l'encodage des caractères en UTF-8 qui est le type d'encodage le plus utilisé puisqu'il prend en charge la majorité des caractères et langues parlés dans le monde

Balise Link pour lier nos feuilles de styles à la page.

Boucle foreach va venir récupérer les fichiers instancier dans le controller qui seront lier uniquement à une page précise

Balise Script pour lier le javascript

Même chose pour la foreach

Slide 16

Je créé le controller de ma page

Le namespace va instancier le chemin de mon fichier

Use va permettre de faire comprendre au controller qu'il doit utiliser le Template vu à la slide juste avant

Je crée une class chiffreController qui prend une fonction et renvoi au Template, grâce à la method render les éléments dont aura besoin le Template.

Slide 17

Dans le fichier vue, il y a l'ensemble de la structure HTML mais je vais vous présenter uniquement la partie qui nous intéresse : la map :

Cette section prend 2 Div : une infobulle qui s'affiche dynamiquement avec les données demandées que nous verrons dans la suite de la présentation et une mappemonde au format SVG.

J'ai choisi ce format car c'est un format puissant qui va me permettre de jouer avec des animations et crée de l'interactivité pour récupérée des données. Il a l'avantage aussi d'être vectorielle et basé sur du XML donc il peut être modifier avec n'importe quel éditeur de code

Parler des class Bootstrap.

Slide 18

Malgré les différentes class Bootstrap, j'ai aussi du CSS personnalisé pour certains éléments

Slide 19

Passons maintenant à la partie la plus intéressante : le Javascript

Dans un premier temps j'instancie l'ensemble des éléments du DOM dont j'aurais besoin dans mon fichier Javascript.

J ’utilise les 2 méthodes d’interface différente pour tout ce qui sera non spécifique et statique, j’utilise une nodeList avec querySelector pour les éléments ciblés et dynamique, j’utilise une htmlCollection avec getElementById

Grace à l’attribut setAttribute, j’ajoute dynamiquement à la class tooltip un attribut .

Grâce à getComputedStyle qui va me donner toutes les propriétés de l’élément ciblé, je récupère sa largeur et sa hauteur que je garde en mémoire avec ma constante.

J’initialise une une nouvelle date qui sera celle du jour uniquement pour pouvoir récupérer l’année grâce à getFullYear.

Pour le bien d’un de mes compteurs j’instancie une date au 1er Janvier de l’année en cours. Avec Année – Index du Mois commençant à 0 et jour

Puis je finis par initialiser l’ensemble des informations par pays. Comme j’ai plusieurs informations par pays, je crée un objet qui me permettra une lecture plus facile grâce au couple clé/valeur

Slide 20

Avant de lancer mes fonctions, je déclare mes setInterval qui nous serviront plus tard.

J’ai créé 2 types de compteur : 1 qui se lance au chargement de la page un autre qui s’active au clic

Pour mener à bien leur activation, j’utilise l’écouteur d’évènements addEventListener qui va « écouter »tout ce qui se passe sur le DOM

Je commence par un un écouteur sur l’évènement « load » qui se charge de lancer ma fonction loopFr qui prend en paramètre des valeurs de mon Objet France.

Le second va écouter l’évènement « clic » qui lancera 2 fonctions distinct.

Ma fonction fléchée commence par récupérer l’id de la cible du clic (d’où l’intérêt d’avoir une carte en SVG).

Puis j’efface les potentiels setInterval en cours qui va permettre de remettre les compteurs à 0

Et maintenant je peux lancé mes fonctions correctement. Ayant plusieurs pays de déclaré, j’ai opté pour un switch case qui prend en argument la cible du clic récupérer juste avant et en fonction de son ID va lancer 2 fonctions avec les paramètres adéquats.

1 Fonction compteur pour les émissions de CO2 d’un citoyen d’un pays

L’autre pour les émissions du pays en question

Slide 21
Si toutefois un pays n’est pas répertorié on entra dans le cas default et j’affiche un message dynamiquement qui explique qu’on n'a pas les informations pour ce pays.

Cet affichage est fait en textContent pour éviter les injections de code malveillant cette fonction traite l’information comme du texte brut.

Pour un affichage optimal, je crée un container qui sera reconnu comme une div, je crée mes éléments de listing. Une fois mes éléments créer je les ajoute au container qui sera ensuite envoyer à l’éléments du DOM que j’ai déclaré dès le départ.

Pour afficher le résultat de mes fonctions j’ai besoin que mon infobulle apparaisse à l’écran donc je commence par récupérer les coordonnées de mon pointeur grâce à clientX et Y qui récupère les coordonnées du pointeur sur ce qui est visible à l’écran et scroll X et Y mesure la distance défilée par rapport au coin supérieur gauche du document entier.

A partir de ces éléments plus les éléments de style (widt et height) récupérés plus tôt je peux positionner exactement mon infobulle.

Slide 22
Je vais vous montrer maintenant une fonction qui est lancé au clic, toutes les fonctions sont basées sur le même principe. Ici la fonction d’émission de C02 par citoyen

Pour avoir un effet « chronomètre » et qu’ici on est sur des chiffre relativement petit (par exemple pour un français, on parle de 0.2g par seconde) je détail entièrement le poids du CO2 émit

Je commence par tout mettre à 0

Ma fonction va me retour une fonction setInterval qui va incrément chaque seconde le poids passé en en paramètre. Puis je l’affiche de la même façon qu’expliqué un peu plus tôt pour des soucis de sécurité

Le résultat vous sera montré à la fin de la présentation.

Slide 23
Passons maintenant au Back de l’application

Je vais commencer par une présentation du MCD/MLD de la conception de la base de données

Ensuite, je vous présenterai comment les tables ont été créées

Et je finirai par une présentation du code pour l'envoi et la réception de messages personnels sur l'application.

Slide 24
Le Modèle de Conception de Données est une méthodologie Merise (Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise) qui consiste à avoir une représentation graphique structuré des données pour faciliter la compréhension de leur gestion.

Elle est définie par différents éléments que l'on peut voir ici :

- L'entité qui représenté un objet sur lequel on peut stocker des informations -> les attributs
- Les attributs sont donc les caractéristiques qui décrivent l'entité. -> Décrire les attributs
- Les associations : c'est la représentation d'une relation entre 2 entités ou plus.
- Pour mieux comprendre cette relation, on utilise des cardinalités qui définissent le nombre d'occurrence d'une entité qui peuvent être associé à une autre entité au travers de cette relation
- Exemple : Ici on a une relation de 1,1 de messages vers users ce qui veut dire qu'un message doit être attribué à au minimum un user et au maximum 1 user. Le minimum de 1 va permettre la suppression en cascade : si on supprime un utilisateur tout ce qui lui sera lui sera supprimé.

Dans l'autre sens on a une relation de 0,n c'est-à-dire qu'un user peut ne pas envoyer de messages mais a peut aussi en envoyer plusieurs

Slide 25
Dès lors que les entités et les relations sont créées, on va pouvoir établir le Modèle Logique de Données qui est une représentation des données où les entités, attributs et relations sont spécifiés de manières détaillées.

Le MLD lui, va se concentrer sur la manière dont les données vont être stockées et organisées.

On va y voir apparaitre :

- Les clés primaires qui identifient de manière unique chaque occurrence de l'entité dans une table de BDD
- Les clé étrangère (Foreign Key) qui, elle indique les clés primaires d'une autre entité ce qui a pour but d'établir la relation entre deux entités.

- Et les autres attributs
- On peut voir que les flèches ont un sens. Ce sens est important pour la compréhension de la relation :
- Une flèche qui pointe vers l'extérieur d'une entité signifie que l'entité a une clé étrangère qui fait référence à la clé primaire d'une autre entité. En gros elle dépend d'un autre
- Une flèche qui pointe vers l'intérieur d'une entité signifie elle qu'une clé étrangère est lié à sa clé primaire = en gros que quelqu'un dépend d'elle

Slide 26
Maintenant que j'ai une vision sur l'architecture de ma base de données, je peux commencer à la créer.

J'ai utilisé le logiciel Workbench qui est un outil graphique de BDD et d'administration mySQL. Très bon pour générer du code SQL pour créer ou modifier des BDD relationnelles

Je commence par créer la base de données

Afin de pouvoir travailler dessus, j'utilise USE qui va me permettre de cibler ma BDD

Je crée mes tables avec leurs attributs (énumérer les attributs)

Ensuite je crée la relation entre elle

Alter table va me permettre de venir modifier une table existante à laquelle je vais ajouter une contrainte pour garantir l'intégrité et la validité de la donnée.

Je L'informe que la contrainte sera une Foreign key et lui précise qu'elle doit faire référence à l'attribut id_users dans la table users.

Slide 27

Un fois le script lancé, on se retrouve avec l'ensemble de nos tables correctement créer

Slide 28

Et plus particulièrement les tables users et messages sur laquelle on aperçoit que la relation passe bien.

Bien.

Revenons maintenant sur notre éditeur de texte afin que je vous présente le développement côté back de l'application.

Slide 29
Vous vous rappelez, tout à l'heure quand je vous ai présenté le Modele MVC, je vous ai expliqué comment cela fonctionne. Je vous ai présenter la vue et le controller dans la partie front.

Voyons maintenant le Model.

Pour rappel, le Model va répertorier l'ensemble des fichiers qui servent à représenter la structure des données et la logique métier de l'application, Les model interagissent avec la BDD.

Je commence mon model en lui donnant un chemin grâce à Namespace,

Je fais appel a d'autre fonction qui me seront utile : BddConnect qui est une fonction déportée de la connexion à la base de données et Users qui est un autre model qui me sera nécessaire pour référencer l'expéditeur et le destinataire des messages.

Je crée ma class User qui héritera des données publiques de ma fonction BDD grâce à la méthode extends

Je déclare l'ensemble de mes attributs qui font références aux colonnes de ma BDD . Je les déclare en Private afin de faire comprendre aux système qu'uniquement la class Messages peut les utiliser. On parle alors d'encapsulation.

J'initialise un constructeur qui va me permettre de créer les données publiques de ma class Users

Je continue avec la déclaration de l'ensemble de mes Getter et setter de chaque attribut qui vont me permettre la récupération, la manipulation et la modification des données. Le typage permet d'un seul coup d'œil à savoir ce qui est attendu en retour (utile pour la sécurité et si quelqu'un d'autre doit intervenir sur le code)

Slide 30

Une fois tout initialiser, on peut rentrer dans le concret !

Je peux créer toutes mes fonctions publiques qui vont interagir avec la BDD

Je commence par la fonction d'ajout de message en BDD qui ne promet rien en retour :void.

J’utilise un bloc Try Catch qui va me servir a référencer les potentielles erreurs que le système peut rencontrer

Le Try prendra l’ensemble du code à effectuer. Si Une erreur apparait, il saute automatiquement dans le bloc Catch arrete immédiatement le process grâce à die et me renvoi l’erreur rencontrée.

Dans mon try donc, je commence par stocker les données récupérer via le formulaire côté front dans mes variables locales.

J’instancie ma requête qui va appeler la fonction connexion hérité de l’extend a BddConnect et va préparer une requête SQL , ici une insertion dans la table messages

Pour passer mes variables locales stockées à ma requête, j’utilise un bindParam qui me permet un traitement des données sécurisé et flexible renforcé par un PDO::Param qui va indiquer la nature de la donnée.

Une fois les données bindé, j’exécute ma requête.

Même principe pour les 2 prochaines fonctions présenté. Il n’y a que la requête qui va changer :

- La première va récupérer tous les messages de l’utilisateur avec une jointure avec la table users pour recupérer le pseudo de l’emetteur du message et on va grouper les messages par leur id pour créer une list
- La seconde va récupérer les messages grâce à l’id de discussion de la même façon pour les afficher.
- Voyons comment ca fonctionne :

Slide 31
Sur le controller, la fonction messageReceive démarre instancie les éléments dont elle aura besoin

indMessages va créer un tableau avec les messages afin de les restitué à la vue

Je lance getAllMessageGroupBy vu juste auparvant, si la fonction renvoi null un message s’affiche comme quoi la messagerie est vide, sinon la vue affichera ça

Slide 32

Un fois que l’on clic sur le bouton

Le système va d’abord clean l’entrée de donnée ici , l’id de la discussion par sécurité .

Ce clean est décomposé en 3 parties :

- Un trim qui va supprimer les espaces blancs au début et à la fin de la chaine de caractères

- Un strip_tag qui va supprimer toutes balises html ou php

- Et un htmlspecialchars qui va convertir tous les caractères spéciaux en entité html &lt pour le signe < par exemple

- Une fois le clean fait, on va décodé l'id et l'envoyer en paramètre de la fonciton getPersonalMess . Ce qui aura pour but d'afficher les messages sur l'interface utilisateur

Slide 33
Une fois le bouton cliqué , on recommence le système clean l'id de discussion passé au bouton et on le décode.

- Si le champ de réponse est rempli le système va cleaner la réponse
- Le système récupère le dernier message du flow de discussion
- Si le l'id du destinataire du dernier message n'est pas l'utilisateur connecté alors le destinataire reste le même (ce qui veut dire que l'utilisateur connecté a envoyé le dernier message)
- Sinon le destinataire du message est l'expéditeur du dernier message
- Le controller ayant la class Message (le model) en parent grâce à un extend le mot clé $this me permet d'accéder à ses fonctions publiques pour y passer mes valeurs.
- On finit par lancer la fonction d'insertion de message dans la base avec les valeurs passé au-dessus puis on refresh la page pour afficher le dernier message.

Slide 34
Pour finir,

Ce projet a été conçu au fur et à mesure que la formation s'est déroulée

J'ai eu quelques obstacles durant la conception de mon code notamment niveau JS ou j'ai eu quelques difficultés avec mes compteurs. Au départ n'ayant pas la connaissance du clearInterval à chaque fois que je lançais un nouveau compteur, il s'affiché au-

dessus du compteur en cours ce qui créer un bug énorme mais à force de recherche je suis tombé sur les informations de ce clearInterval et tout est rentré dans l'ordre

L'affichage des pseudos dans la messagerie m'a valu aussi quelques cheveux blancs car à la base je ne récupérer uniquement que l'id qui est accessible dans la table messages puis je me rappelle un matin avoir eu le retour en mémoire de mes cours de SQL et du inner-joint ce qui a réglé mon problème en 30 secondes

En relisant certaine partie de mon code, et en ayant gagné en expérience, je me rends compte qu'il y a des éléments que je ne ferai plus de la même façon aujourd'hui surtout sur l'écriture du code et sa compréhension. Il est tout à fait possible de le simplifier pour le rendre plus lisible.

Et si c'était à refaire, je recommencerai tout à zéro et passerai par React et Remix et un ORM comme Prisma que j'ai découvert durant mon stage ce qui offre des possibilités énormes.