



21 Mai 2023

Projet de programmation

AgiWeb

AITTAHAR | AMAR | BONZOM | CASTA | DEBACHA | FAYNOT | KEVORKIAN | TURPIN



Notre démarche

Introduction &
contexte

01

Base de données

04

Organisation

02

Design

05

Architecture

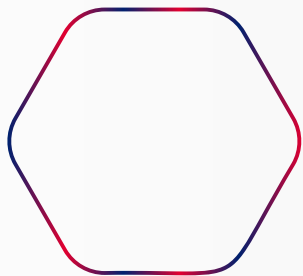
03

Python & FLASK

06

Conclusion

07



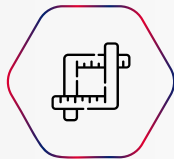
01

Introduction & contexte

InnovaLog



Run 1



Run 2-3



Run 4



Découverte

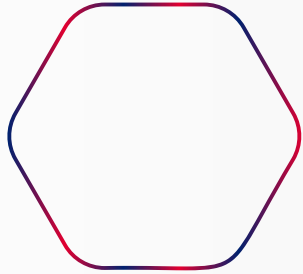
Prise en main du
Serious Game

Chantiers

Besoin d'une plateforme
de suivi de commande

AgiWeb

Implémentation de AgiWeb
dans le Serious Game



02

Organisation

Notre équipe

Designers HTML & CSS

Robin AMAR
Nadia AITTAHAR



Chef de projet

Thomas CASTA



Programmeurs Python & Flask

Max TURPIN
Guillaume FAYNOT



Experts Métier & BDD

Lili BONZOM
Maureen KEVORKIAN
Sabrina DEBACHA



Les rôles

Chef de projet : proposer une méthodologie de travail et une organisation claire afin de faire avancer le projet en respectant les deadlines.

Programmeurs Python-FLASK : mettre en place la liaison entre le code HTML et la BDD grâce à un programme script.

Programmeurs HTML-CSS : réaliser le design complet du site, création des différentes pages ainsi que des formulaires.

Experts Métier et BDD : construire une base de données exploitable à partir des données du serious game, tester le site web et conseiller les programmeurs.

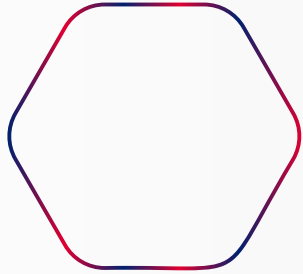
Les outils



- Mutualisation des fichiers BDD, Python, HTML
- Travail en simultané (push)
- Historique des modifications



- Travailler en simultané sur les livrables
- Mise à jour du diagramme de Gantt



03

Architecture du site

Architecture du site

Accueil : la page d'accueil pour le site, proposant une rubrique “En savoir plus”.

Espace Client : une page “Nouvelle commande” donnant accès au choix du châssis puis des options, et une page “Suivi de commande” donnant le récapitulatif des commandes passées.

Espace AgiLean : une page “Commande de kits” permettant de commander des kits de pièces auprès de AgiLog, et une page “Commandes client” donnant le récapitulatif des commandes du client et permettant de les traiter.

Espace AgiLog : une page “Commande de pièces” permettant de commander des pièces, et une page “Affichage des stocks” donnant le récapitulatif des stocks actuels.

Accueil



Espace Client

AGIWEB

ACCUEIL

CLIENT

AGILEAN

AGILOG

Suivi de commandes client

ID Commande	Date Commande	Chassis	Option
1	0:9:27	CCO	Attache-Remorque

Espace AgiLean

AG

Suivi de commandes de pièces AgiLog

AGILOG

ID Commande	Date Commande	Pièce	Quantité	Statut
1	0:9:27	Attache-Remorque	OK	En cours

État de la commande AgiLog

ID COMMANDE

☐

Commande reçue

☐









Commande en attente

☐

Annuler la commande

OK

Espace AgiLog

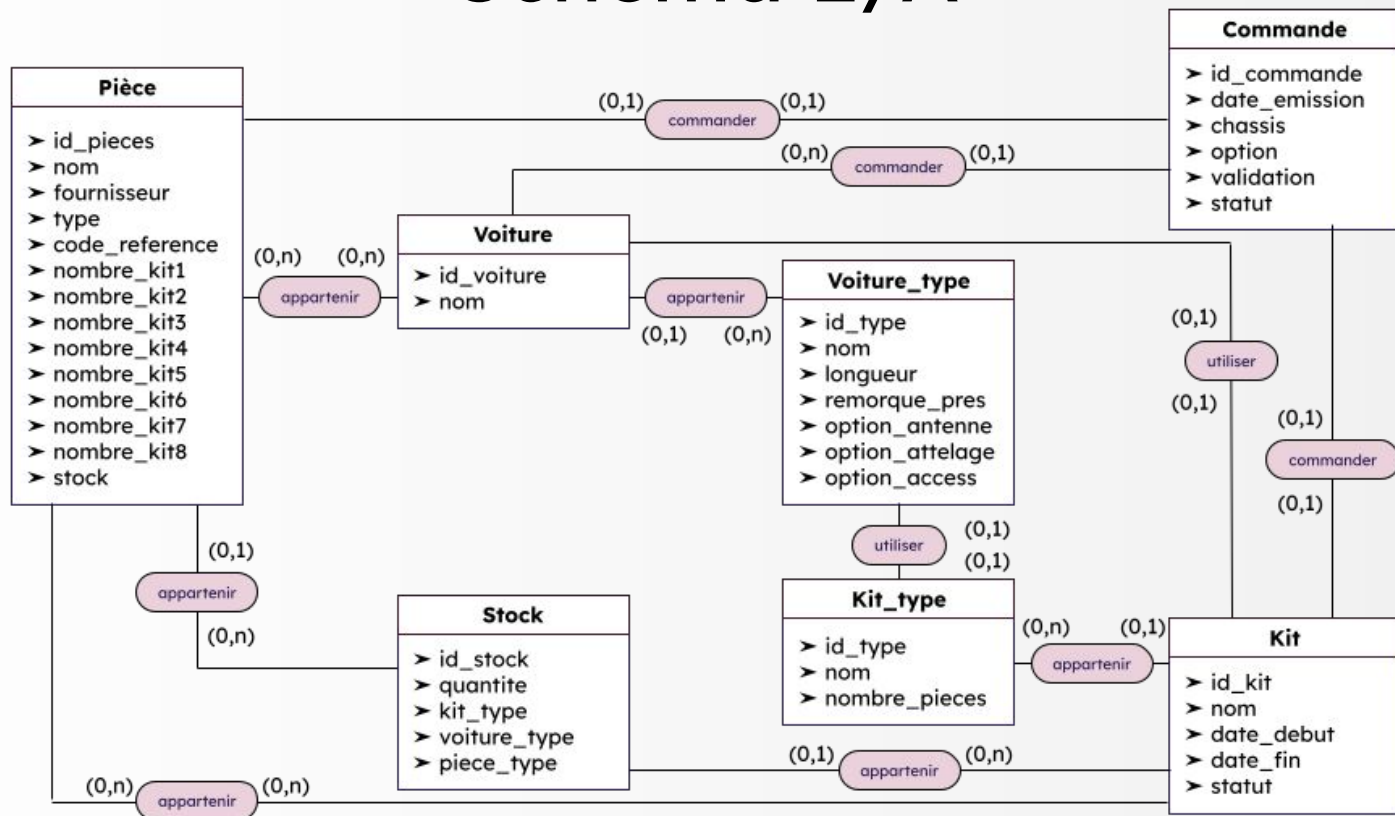
AGIW	PIÈCE	DESCRIPTION	QUANTITÉ	AGILOG
		ARCEAU	<input type="text" value="0"/>	
		ATTACHE	<input type="text" value="0"/>	
		BRIQUE 1X4	<input type="text" value="0"/>	
		BRIQUE 2X1	<input type="text" value="0"/>	
		BRIQUE 2X2	<input type="text" value="0"/>	
		ÉQUERRE	<input type="text" value="0"/>	
		FEU AVANT	<input type="text" value="0"/>	
		FEU ARRIÈRE	<input type="text" value="0"/>	

04

















































Base de
données



Schéma E/A



Aperçu des tables

▼  Commandes	▼  Stock	▼  Pieces
 id_commande	 id_stock	 id_pieces
 date_emission	 quantite	 nom
 Chassis	 kit_type	 fournisseur
 Option	 voiture_type	 type
 validation	 pieces_type	 code_reference
 statut		 nombre_Kit1
▼  Kit_type	▼  Voiture	 nombre_Kit2
 id_type	 id_voiture	 nombre_Kit3
 nom	 nom	 nombre_Kit4
 nombre_pieces	▼  Voiture_type	 nombre_Kit5
▼  Kits	 id_type	 nombre_Kit6
 id_kits	 nom	 nombre_Kit7
 nom	 longueur	 nombre_Kit8
 date_début	 remorque_presence	
 date_fin	 options_antennes	
 statut	 options_attelage	
	 options_accessoires	

Requêtes en Python

#INSERT INTO

a= fonction en python pour avoir l'heure de la commande

b=Si c'est CLO, CCO etc...

c= Le nom de l'option ou les options [text]

d= « Ok »

f= « En cours »

INSERT INTO Commandes (date_emission, Chassis, Option, validation, statut)

VALUES ("a","b","c","d","f")

NB : id_commande se créait tout seul

#UPDATE

"UPDATE Commandes SET statut= ? WHERE id_commande=?", (Statut, num_commande)

#SELECT

SELECT MAX(id_commande) FROM Commandes WHERE Chassis='CLO' OR Chassis='CLO' OR Chassis='CLF'

"SELECT Chassis FROM Commandes WHERE id_commandes=" str(num_commande)

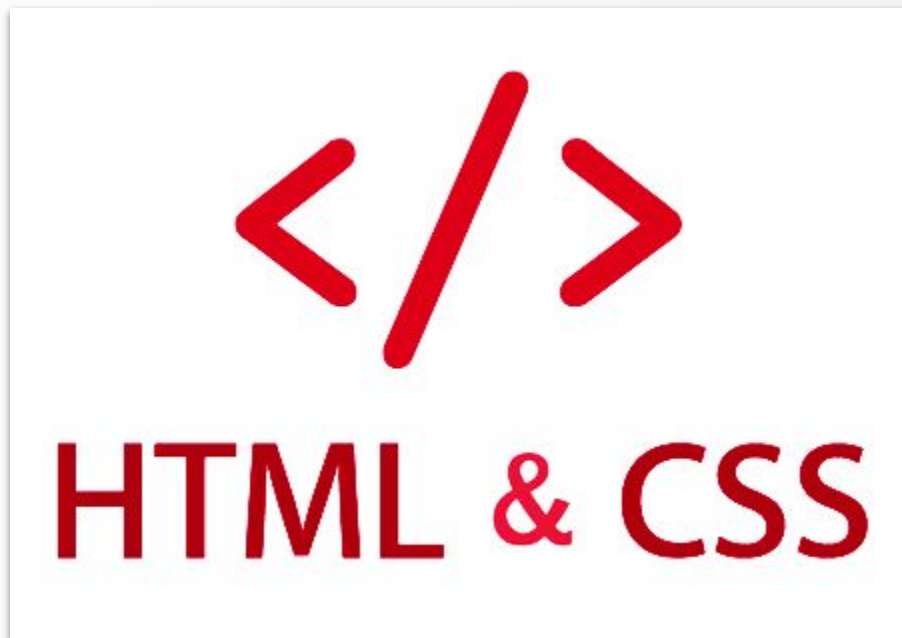
The background features a dark purple gradient on the left, transitioning into a white area on the right. Large, stylized red and purple geometric shapes, including hexagons and curved lines, are scattered across the composition.

05

Design

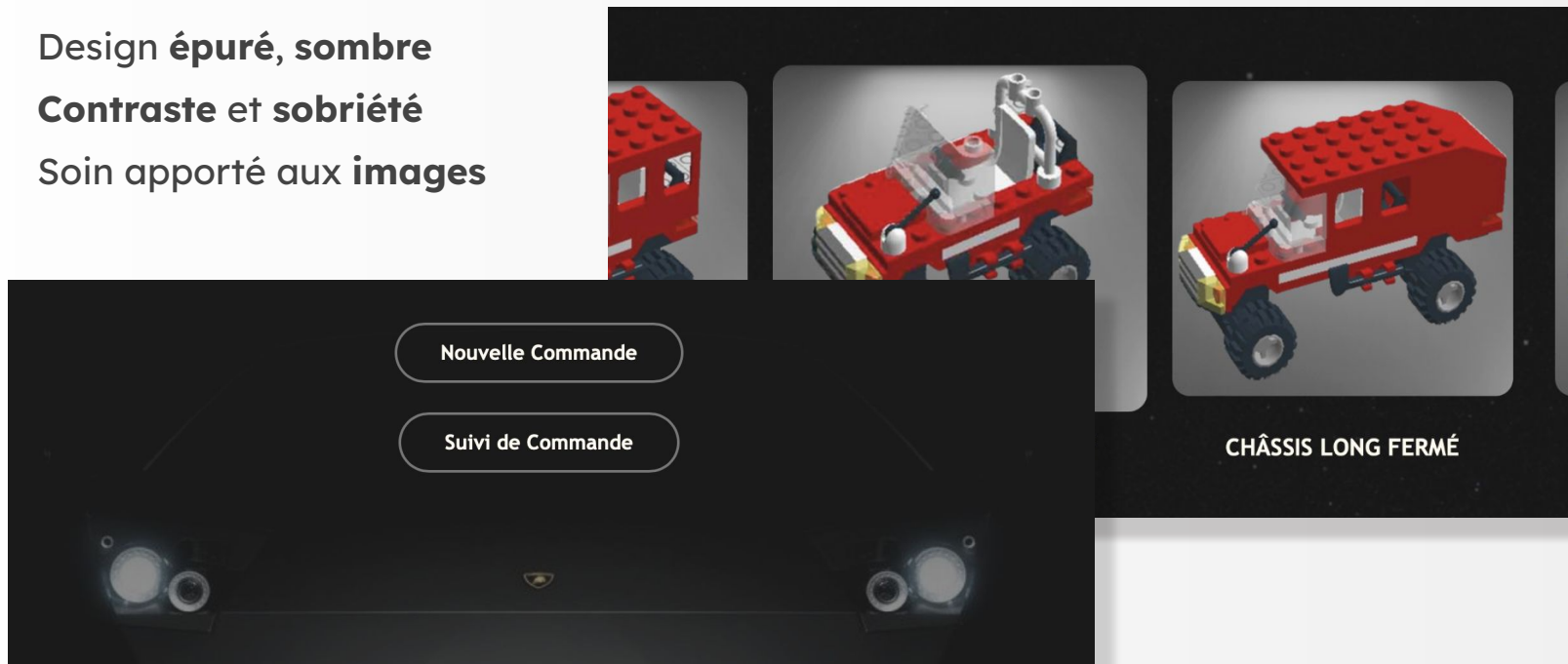
HTML & CSS

Premiers pas en HTML & CSS



Apparence globale

Design **épuré, sombre**
Contraste et sobriété
Soin apporté aux **images**



Code HTML — CSS

```
1 ▼ *{
2   margin:0;
3   padding:0;
4   font-family: "Trebuchet MS";
5 }
6
7 ▼ .banner{
8   width: 100%;
9   height: 135vh;
10  background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0.93), rgba(0, 0, 0, 0.1)),
11  url(2.jpg);
12  opacity: 0.9;
13  background-size: cover;
14  background-position: center;
15 }
16
17 ▼ .logo{
18   width: 300px;
19 }
20
21
22 ▼ nav{
23   width: 90%;
24   margin: auto;
25   padding: 35px 0;
26   display: flex;
27   align-items: center;
28   justify-content: space-between;
29 }
30
```

Animations et transitions



```
47
48 ▼ .navbar ul li::after{
49     content: "";
50     height: 5px;
51     width: 0;
52     background:#FFFAF0;
53     /* barre du centre vers l'extérieur :*/
54     display: block;
55     margin: auto;
56     transition: 0.5s;
57 ▼ /*barre de gauche à droite :
58     position: absolute;
59     left: 0;
60     bottom: -3px;
61     transition: 0.5s;*/
62 }
63
64 ▼ .navbar ul li:hover::after{
65     width: 100%;
66 }
```

The background features a vibrant red-to-purple gradient. On the left, there are several hexagonal shapes with color gradients (pink-to-blue, red-to-blue) and a large, stylized white wavy line that curves across the frame.

06

Programmation

Python & FLASK

Fonctionnement du script

```
AGIWEB > script.py > NouvelleCommandecf
1  from flask import Flask, request, url_for, render_template, session
2  import sqlite3 as lite
3  import time as t
4
5  # -----
6  # application Flask
7  # -----
8
9  app = Flask(__name__)
10 app.secret_key = 'secret'
11
12 # RENSEIGNER L'ADRESSE DE LA BDD
13 BDD=r"\\Base_de_donnees_Serious_Game.db"
14 t0=t.time()
15
16 # Renvoie str de heures, minutes, secondes depuis le lancement du run
17 > def temps():...
18
19 ##### Accueil #####OK
20 @app.route('/')
21 def Accueil():
22     return render_template('index.html')
23
24 ##### Client #####OK
25 @app.route('/Client')
26 def Client():
27     return render_template('lclient.html')
28
29 ##### Client/Nouvelle Commande #####OK
30 @app.route('/Client/NouvelleCommande')
31 def NouvelleCommande():
32     return render_template('lcmde.html')
33
34
35
```

Extraits du code

```
##### Client/Nouvelle Commande/repcl #####OK
@app.route('/Client/NouvelleCommande/repcl', methods=['GET','POST'])
def NouvelleCommanderepcl():
    # Données des checkboxes de type str()
    Option = ', '.join(request.form.getlist('option'))
    # Chercher le code chassis
    Chassis = session.get('chassis')
    message = f"Merci, votre commande a bien été prise en compte."
    # Insertion BDD
    con = lite.connect(BDD)
    con.row_factory = lite.Row
    cur=con.cursor()
    cur.execute("INSERT INTO Commandes(date_emission, Chassis, Option, validation, statut) VALUES (?, ?, ?, ?, ?)")
    con.commit()
    con.close()
    return render_template('1repcl.html', message = message)
```

Extraits du code

```
##### Agilean/ Commande client #####OK
@app.route('/Agilean/CommandeClient', methods=['GET','POST'])
def AgileanCommandeClient():
    # Affichage des commandes clients
    # Connexion BDD
    con = lite.connect(BDD)
    con.row_factory = lite.Row
    cur=con.cursor()
    # Affichage des commandes en cours
    cur.execute("SELECT id_commande,date_emission, Chassis, Option, validation, statut FROM Commandes WHERE
lignes=cur.fetchall()
cur.execute("SELECT id_kits,nom,date_début,Statut FROM Kits")
lignes2=cur.fetchall()
return render_template('1sclnt.html',commandes=lignes,kits=lignes2)
```

Extraits du code

```
##### AgiLog/Commande de pièces/replo #####
@app.route('/AgiLog/CommandePièce/replo', methods=['GET', 'POST'])
def AgiLogCommandePiecereplo():
    con = lite.connect(BDD)
    con.row_factory = lite.Row
    cur=con.cursor()
    # Liste contenant les pièces commandées
    Pieces=[]
    qteliste=['qtea','qteb','qtec','qted','qtee','qtef','qteg','qteh','qtei','qtej','qtek','qtel','qtem','qten','qteo','qtep','qteq','qter','qtes','qteu','qtev','qtey','qtez']
    for qtetruc in qteliste:
        qp=request.form[qtetruc]
        if qp=='':
            qp=0
        Pieces.append(int(qp))
    # MAJ des stocks pour chaque pièce
    for i in range(len(Pieces)):
        if Pieces[i] != 0 :
            cur.execute("UPDATE Pieces SET stock=stock+? WHERE id_pieces = ?", (Pieces[i],i))      #On modifie la quantité de stocks
    con.commit()
    con.close()
    message = f'Merci, votre commande a bien été prise en compte.'
    return render_template('1replo.html',message=message)
```



07

Conclusion



Merci pour votre attention !