

PROJET ARGILES

GUIDE DEVELOPPEUR SITE WEB

Besoin d'aide ?

Si vous avez des questions, contactez Thomas Lebarbé via son email de l' [Université Grenoble Alpes](#).

Page d'accueil

Ce document décrit le fonctionnement et la structure du code de la page d'accueil du projet ARGILES. Il vise à faciliter les futures modifications et améliorations du projet.


La page d'accueil est construite en HTML, CSS et JavaScript avec l'aide de **Bootstrap** pour le design responsive. Elle comprend plusieurs sections clés :

- ❖ Un menu latéral interactif (masqué par défaut).
- ❖ Des liens vers les autres pages du site.
- ❖ Un design dynamique avec des effets CSS (zoom sur images, arrière-plan, animations SVG).
- ❖ Des informations sur les œuvres étudiées et sur le projet.

Fichiers utilisés :

- ❖ [index.html](#) : Contient la structure principale de la page.
- ❖ [img/](#) : Dossier contenant les images des livres et les pdfs du projet.
- ❖ [wave.svg](#) : Fichier SVG utilisé pour l'effet de vague en bas des pages.

1. Le menu latéral

Le menu latéral est caché par défaut et s'affiche lorsqu'on clique sur le bouton  Menu. Il permet la navigation vers d'autres pages du site.

Code HTML

Le menu latéral est défini avec la classe `.sidebar` et contient des liens vers les autres pages avec `a href="nom_page.php/.html">Nom Page`

Code CSS

Le menu latéral est masqué et s'affiche grâce à la transition. Mise en page dans la section `<style> </style>`.

JavaScript

La fonction `toggleSidebar()` permet d'afficher/masquer le menu.

2. Contenu de la page

La section principale de la page contient plusieurs éléments :

- ❖ Une présentation du projet ARGILES.
- ❖ Une explication des classes scolaires.
- ❖ Les œuvres étudiées et les questions posées aux élèves.
- ❖ Des images interactives avec un effet de zoom.

Code HTML

La structure de la page est organisée en sections `<section>` et paragraphes `<p>` pour une meilleure lisibilité. Exemple:

Code CSS

Les titres et les textes ont été stylisés pour une meilleure lisibilité.

```
h1, h2, h3, h4 {
```

```

    color: #2c3e50;
}
body {
    font-size: 18px;
    line-height: 1.6;
}

```

Effet de zoom sur les images

Les images s'agrandissent lorsqu'on passe la souris dessus.

```

.image-hover {
    transition: transform 0.3s ease-in-out;
}
.image-hover:hover {
    transform: scale(1.3);
}

```

3. Accès aux autres fonctionnalités

La page d'accueil offre un accès rapide aux autres fonctionnalités du projet via le menu latéral.

Pages associées :

- [database_visualization.php](#) → Affichage dynamique du contenu des fichiers XML ajoutés à la base de données.
- [pdf_visualization.php](#) → Consultation des copies manuscrites des élèves.
- [upload.html](#) → Formulaire de téléversement de nouveaux fichiers XML dans la BDD : accessible uniquement aux usagers autorisés grâce à la page [login.php](#).
- [functions.html](#) → Carte interactive du site.

4. Design dynamique : effets de fond et décorations

Le fond de la page est un dégradé de rose et vert pétrole, avec une légère opacité pour la lisibilité.

```
body {  
  background: linear-gradient(160deg, #d8a7b1 10%, #2f5d62 90%);  
}
```

Un fichier SVG est utilisé pour un effet de vague décoratif en bas de la page. Le fichier se trouve dans le dossier `img`.

```
<div class="wave"></div>
```

```
.wave { code CSS pour la vague }
```

5. Améliorations Possibles pour les Futures Versions

Actuellement, la page est optimisée pour les ordinateurs. Une amélioration pourrait inclure une meilleure mise en page responsive pour les mobiles et tablettes. On pourrait aussi ajouter des animations CSS plus fluides et personnaliser davantage le design du menu latéral ou l'affichage des images.

Page de visualisation de la base de données

Les scripts concernés :

- ❖ `Database_visualization.php`
- ❖ `Filter.php`
- ❖ `Tableau_accueil.php`

Ils sont conçus pour assurer la visualisation et l'exploration dynamique des données contenues dans la base de données du projet ARGILES. Ils permettent d'afficher les résultats sous forme de tableau interactif avec des options de filtrage avancées, de lier les réponses individuelles des élèves à des pages d'analyse approfondie et d'exécuter des statistiques sur la fréquence des mots utilisés.

Les principaux composants de ce système sont :

- Un système de filtrage dynamique permettant d'afficher uniquement les informations pertinentes.
- Une table interactive affichant les résultats et permettant une exploration intuitive.
- Une liaison entre PHP et Python pour générer les résultats d'analyse approfondie.

1) Explication du fonctionnement des scripts PHP

Fichier de connexion à la base de données

Tous les scripts incluent `connexion.php` afin d'éviter la duplication du code de connexion. Ce fichier contient les informations nécessaires à la connexion à la base de données.

```
include_once("../connexion.php");
```

Cette approche permet de centraliser la connexion et de faciliter la mise à jour de l'accès à la base de données.

Récupération et traitement des requêtes SQL

a) Récupération des données

La variable `$_POST['req']` contient la requête SQL envoyée par l'utilisateur via l'interface.

```
$query = $_POST['req'];
```

La requête est ensuite exécutée sur la base de données :

```
$result = $db->query($query);
$vars = $result->fetchAll(PDO::FETCH_ORI_FIRST);
$vars = array_keys($vars[0]); // Récupération des noms des colonnes
```

b) Affichage des erreurs et vérification

Si aucune donnée n'est sélectionnée, un message est affiché :

```
if ($requete == 'NONE') {
    print "<div class='table_main'>
        <div id='no_data'>
            Pas de données sélectionnées.
            Veuillez sélectionner au moins un ouvrage, un type de question et un groupe.
        </div>
    </div>";
}
```

Gestion des filtres dynamiques

Le système de filtres permet aux utilisateurs de sélectionner des critères de recherche pour affiner les résultats affichés dans le tableau.

Les filtres disponibles sont :

- ❖ Par ouvrage (chargé depuis un fichier `db_list.txt` listant les ouvrages disponibles)
- ❖ Par type de question (like, dislike, expectation, evocation)
- ❖ Par groupe d'élèves
- ❖ Par colonnes visibles

Chaque filtre est généré dynamiquement en parcourant la base de données.

Exemple pour les ouvrages :

```
<?php foreach($db_list as $oeuvre) {
    $id_oeuvre = explode('|',$oeuvre)[0];
    $nom_oeuvre = explode('|',$oeuvre)[1];
    print '<div>
        <input type="checkbox" id=".'.$id_oeuvre.'" name=".'.$nom_oeuvre.'"
class="ouvrage" checked onChange="javascript:update(this)"/>
        <label for=".'.$nom_oeuvre.'">'.$nom_oeuvre.'</label>
    </div>';
}
?>
```

Affichage des résultats dans le tableau dynamique

Chaque colonne du tableau est créée dynamiquement en fonction des résultats SQL.

```
foreach(array_keys($vars) as $i) {
    print '<div class="table_header_col cell_'.$i.'" style="padding:0px">';
    print '<div class="table_header_cell_title">'.$vars[$i].</div>';

    $reponse = $db->query($requete);
    $ligne = $reponse->fetchAll();
    foreach($ligne as $cell) {
        print "<div class='table_cell question_". $cell[$index_type]."'>";
```

```

        print $cell[$i];
        print "</div>";
    }
    print '</div>';
}

```

Gestion de la mise en page et de l'affichage dynamique

Le projet intègre : un affichage adapté via Bootstrap et CSS personnalisé; une interaction fluide grâce à AJAX et JavaScript pour charger dynamiquement les filtres et tableaux; une structuration des données facilitée par des requêtes SQL optimisées

2) Intégration avec les scripts Python pour l'analyse NLP

Deux intégrations essentielles permettent d'exploiter des analyses NLP via des scripts Python :

- Affichage des résultats d'analyse d'un ID d'élève ([get_audite_results.php](#))
- Affichage des statistiques de fréquence des mots ([stats_results.php](#))

Lorsqu'un ID élève est cliqué, il renvoie vers [get_audite_results.php](#) en passant l'ID et l'ouvrage en paramètres GET :

```

if ($i == $index_id) {
    print "<a
href=\".$url.\"/get_audite_results.php?audite_id=\".$cell[$index_idAudite].\"&novel_title=\".$
cell[$index_oeuvre].\" target='_blank'>";
}

```

De la même manière, une analyse statistique des textes des réponses par chaque ouvrage est lancée en accédant à [stats_results.php](#). Ces scripts fournissent un système puissant et dynamique pour explorer et analyser les réponses des élèves. L'intégration de PHP et Python permet une exploitation avancée des données tout en maintenant une interface intuitive et rapide.

3) Améliorations possibles

- ❖ Optimisation des requêtes SQL : réduire le nombre de requêtes répétées dans les filtres.
- ❖ Ajout d'un mode "exportation" pour sauvegarder les résultats des requêtes en CSV..
- ❖ Mise en cache des résultats pour accélérer l'affichage des données filtrées.

Page de visualisation des copies manuscrites

Cette page permet une visualisation claire et structurée des copies d'élèves. Cependant, l'ajout d'éléments interactifs (navigation améliorée, adaptation mobile et chargement dynamique des PDFs) améliorerait considérablement l'expérience utilisateur et la maintenance du site.

Structure HTML

- ❖ Le `<header>` contient le titre de la page, une courte description et un bouton de retour.
- ❖ La `<div class="main-content">` organise les différentes sections de PDF en fonction des classes.
- ❖ Le `<footer>` affiche les informations légales et les contacts du projet.

Structure CSS (mise en page)

- ❖ Fond dégradé : la page reprend la charte graphique du projet avec un dégradé (`d8a7b1` → `2f5d62`).
- ❖ En-tête et pied de page assurent une cohérence visuelle avec le reste du site.
- ❖ Menu latéral interactif qui s'affiche en cliquant sur un bouton grâce à un effet de transition (comme déjà décrit, voir page d'accueil).
- ❖ Visionneuses PDF intégrées (`.pdf-container iframe`) avec une taille fixe.

JavaScript

La fonction `toggleSidebar()` permet d'ouvrir et fermer le menu latéral.

Améliorations possibles

→ Meilleure adaptation aux écrans mobiles

La page n'est pas entièrement optimisée pour les petits écrans (téléphones et tablettes).

→ Affichage dynamique des PDFs

Actuellement, les fichiers PDF sont déclarés manuellement dans `(src="img/siriuspdf/4-G1_Sirius.pdf")` et stockés dans un dossier images. On pourrait utiliser du JavaScript pour charger dynamiquement les fichiers PDF depuis un dossier, afin de faciliter l'ajout de nouvelles copies.

→ Amélioration de la navigation dans les PDFs

Actuellement, la seule manière d'explorer un PDF est de faire défiler verticalement. Il est envisageable d'ajouter des boutons "Page suivante / Page précédente" pour une navigation plus fluide.

→ Ajout d'une option plein écran

La taille actuelle des PDF (`max-width: 800px; height: 500px;`) peut limiter la lisibilité. Il serait possible d'intégrer un bouton plein écran permettant d'afficher le PDF en taille réelle.

Pages de visualisation des traitements NLP

Fichiers concernés:

- ❖ `get_audite_results.php`
- ❖ `stats_results.php`

Ces pages sont des interfaces permettant de récupérer et d'afficher les résultats des scripts Python `get_audite.py` et `stats.py`, respectivement. Ces pages ne sont pas accessibles directement via la navigation du site, mais sont intégrées à la page de visualisation de la base de données (`database_visualization.php`). L'utilisateur accède à ces pages en cliquant sur les idAudite de la base de données qui sont dynamiquement liées aux scripts Python pour avoir l'analyse détaillée (close reading) et sur le bouton Statistiques pour avoir les analyses plus larges (distant reading). La mise en page de ces pages est réglée par des fichiers CSS externes : `table.css` et `python_style.css`.

1. Fonctionnement général des pages PHP

Ces pages suivent une structure commune :

- Récupération des paramètres GET : les identifiants des personnages ou des livres sont envoyés via l'URL depuis la page de visualisation.
- Exécution du script Python correspondant : en fonction des paramètres fournis, le script Python est exécuté via `shell_exec()`.
- Traitement des résultats : les sorties des scripts Python sont analysées et affichées sous un format structuré.
- Affichage des résultats avec un style cohérent : chaque page suit la charte graphique du site (header/en-tête, pied de page/footer, menu latéral/sidebar, fond dégradé, vague en bas de page, texte adapté pour améliorer la lisibilité, etc.).

2. Détails sur `get_audite_results.php`

Cette page permet d'obtenir l'analyse des réponses des élèves concernant un personnage spécifique d'un roman.

Entrées (Paramètres GET)

- `audite_id` : Identifiant du personnage.
- `novel_title` : Titre du roman.

Processus

1. Vérification des paramètres GET et affichage d'une erreur en cas d'entrée invalide.
2. Exécution du script Python `get_audite.py` avec les paramètres récupérés.
3. Récupération de la sortie du script et formatage des résultats.
4. Affichage des données sous un format structuré (tableau coloré).
5. Affichage des idées en entier.

3. Détails sur `stats_results.php`

Cette page génère une analyse statistique des réponses des élèves pour chaque ouvrage, notamment l'analyse des mots les plus fréquents dans les différentes catégories (évocation, adhésion, rejet, attentes).

Entrées (Paramètres GET)

- `novel_title` : Titre du roman.
- `min_id` : ID minimum du personnage.
- `max_id` : ID maximum du personnage.

Processus

1. Vérification des paramètres GET.

2. Exécution du script Python `stats.py` avec les paramètres récupérés.
3. Récupération et affichage des statistiques sous forme de tableau.
4. Possibilité d'intégration future d'une visualisation graphique.

4. Améliorations possibles

- ❖ Affichage dynamique : Ajouter des visualisations graphiques (nuages de mots, histogrammes) pour améliorer la compréhension des données.
- ❖ Gestion d'erreurs améliorée : Vérifier plus finement les erreurs liées aux scripts Python (exécution, format de sortie).
- ❖ Optimisation des performances : Mettre en cache les résultats des analyses pour éviter d'exécuter les scripts Python à chaque requête.

5. Accès et intégration

Ces pages ne sont pas directement accessibles par un menu, mais intégrées dynamiquement à la page de visualisation de la base de données. Pour tester leur bon fonctionnement :

1. Accéder à `database_visualization.php`.
2. Cliquer sur une valeur pertinente pour lancer l'analyse correspondante.
3. Vérifier si la redirection vers `get_audite_results.php` ou `stats_results.php` fonctionne correctement avec les paramètres GET.
4. Vérifier que les résultats sont bien formatés et correspondent aux attentes.

Pages d'upload des XML : accès restreint aux utilisateurs autorisés

Données de connexion stockées dans la base de données:

```
$valid_username = "argiles"; // Choisir le nom utilisateur
```

```
$valid_password = "#ProjetProIDL2425"; // Choisir le mot de passe
```

Fichiers concernés :

Login.php → Vérifie l'authentification et redirige vers [upload.php](#)

Upload.php → Interface de téléversement et envoi vers [upload_xml.php](#)

Upload_xml.php → Traitement des fichiers [XML](#) et insertion dans [MySQL](#)

1. Connexion et gestion des utilisateurs ([login.php](#))

- ❖ Ce fichier permet aux utilisateurs de s'authentifier avant d'accéder à l'espace sécurisé.
- ❖ Les identifiants sont stockés dans la base de données.
- ❖ Si l'authentification réussit, une session PHP est créée ([\\$_SESSION\["authenticated"\] = true](#)).
- ❖ En cas d'échec, un message d'erreur s'affiche.

2. Page de téléversement ([upload.php](#))

- Vérifie que l'utilisateur est bien connecté via [\\$_SESSION\["authenticated"\]](#).
- Contient un formulaire avec :
 - ◆ Un bouton de sélection de fichier.
 - ◆ Un affichage dynamique du fichier sélectionné.
 - ◆ Un bouton de téléversement qui envoie le fichier xml vers [upload_xml.php](#) pour les traitements et l'ajout dans la base de données.

- Affiche un message de rappel sur le format de nommage des fichiers.
- Contient un bouton de déconnexion redirigeant vers [login.php](#).

3. Traitement des fichiers XML ([upload_xml.php](#))

1) *Vérification de l'authentification*

Si l'utilisateur n'est pas authentifié, il est redirigé vers [login.php](#).

2) *Connexion à la base de données*

Connexion à **MySQL** avec [mysqli](#) ([argiles](#)) et vérification de la connexion avant de poursuivre.

3) *Création des tableaux s'ils n'existent pas*

[Sirius](#) et [RomeoJuliette](#) sont créées pour stocker les données des fichiers XML. D'autres tableaux pourraient être créés en futur pour stocker les informations d'autres ouvrages.

4) *Vérification et traitement du fichier XML*

Vérifie que l'extension est bien [.xml](#). Charge le fichier via [simplexml_load_file\(\)](#).

Extrait les informations suivantes :

- ❖ [idEtude](#) : Identifiant de l'étude
- ❖ [sujet](#) : Titre de l'ouvrage (utilisé pour choisir le tableau cible)
- ❖ [idQuestion](#) : Identifiant de la question
- ❖ [texteQuestion](#) : Texte de la question
- ❖ [idGroupe](#) : Identifiant du groupe (classe)
- ❖ [nomGroupe](#) : Nom de la classe
- ❖ [idAudite](#) : Identifiant de l'audité
- ❖ [idIdee](#) : Identifiant de l'idée
- ❖ [ideeView](#) : Une aperçue du texte de l'idée exprimé par l'élève

- ❖ Insère les données dans le tableau **correspondante** ([Sirius](#) ou [RomeoJuliette](#)) dans la base de données. Chaque tableau aura ses propres indices pour identifier les données insérées.
- ❖ Si l'ajout des données est positif, l'utilisateur est automatiquement redirigé vers [database_visualization.php](#) pour visualiser les données insérées et les explorer grâce aux filtres et aux traitements NLP.

Attention: le format XML du projet doit être respecté pour pouvoir extraire les informations correctement.

4. Structure des tableaux dans MySQL

Colonne	Type	Description
id	INT (AUTO_INCREMENT)	Identifiant unique
idEtude	INT	Identifiant de l'étude
sujet	VARCHAR(255)	Sujet de l'étude
idQuestion	INT	Identifiant de la question
texteQuestion	TEXT	Contenu de la question
type	VARCHAR(50)	Type de la question

idGroupe	INT	Identifiant du groupe/classe
nomGroupe	VARCHAR(255)	Nom du groupe/classe
idAudite	INT	Identifiant de l'audité
idIdee	INT	Identifiant de l'idée
ideeView	TEXT	Contenu de l'idée: une aperçue

5. Améliorations techniques et esthétiques possibles

S'assurer que les fichiers XML respectent bien le format attendu selon le projet (ça avait été fait d'avance pour nos fichiers et on n'avait pas les informations de formatage des XML).

Amélioration de l'interface utilisateur :

- ❖ Ajouter une barre de progression lors du téléversement.
- ❖ Permettre le téléversement de plusieurs fichiers XML en une seule fois.

Page des fonctionnalités : une carte du site

1. Structure du Code

La page est un simple **HTML** constituée de plusieurs sections clés qui permet uniquement d'avoir un accès direct à d'autres pages et serait surtout intéressant lorsque le site aura d'autres pages en futur.

→ En-tête (**<header>**)

- ◆ Contient le titre et la description du projet.
- ◆ Inclut un bouton de retour à l'accueil.
- ◆ Largeur ajustée pour s'adapter au reste de la mise en page.

→ Menu Latéral (**.sidebar**)

- ◆ Initialement masqué, il apparaît lorsqu'on clique sur le bouton **Menu**.
- ◆ Contient des liens vers les principales pages du site.
- ◆ Caché par défaut.

→ Navigation par Bulles (**.bubble-container**)

- ◆ Affiche les fonctionnalités sous forme de **bulles interactives**.
- ◆ Chaque bulle redirige vers une autre page du site via un lien ****.
- ◆ Le design des bulles est dynamique avec un effet de zoom au survol.

→ Pied de Page (**<footer>**)

- ◆ Contient les informations légales et les crédits.
- ◆ Affiche un lien vers la licence du projet.

2. Fonctionnalités Techniques

- 1) Affichage et masquage du menu latéral : géré via une fonction JavaScript **toggleSidebar()**, qui change la position du menu latéral.
- 2) Dans la section **<style> ... </style>** de la page vous pouvez apporter des modifications visuelles pour modifier la mise en page de tous les éléments et

l'aspect global de la page. Cette section peut aussi être exportée comme un **CSS** séparé en futur si nécessaire.

3. Points d'amélioration possibles

- ❖ Intégration d'animations supplémentaires sur les bulles.
- ❖ Ajouter d'autres fonctionnalités dans le site.
- ❖ Amélioration de l'accessibilité et une meilleure compatibilité avec les différents écrans.