# Présentation du projet

# ArticleHub

Présenté par Esteban Bare
Titre Professionnel Développeur Web Et Web Mobile

# Introduccion

ArticleHub est une plateforme web dédiée à la publication, la lecture et l’interaction autour d’articles. Ce projet vise à offrir aux utilisateurs une expérience riche et engageante, où ils peuvent découvrir du contenu de qualité, interagir avec les auteurs, et contribuer par leurs propres écrits. Le site se veut intuitif, accessible et dynamique, favorisant une communauté active et impliquée.

# Concepcion et Codage

# Objectif

Mon projet vise à permettre aux utilisateurs de créer, lire, aimer et commenter des articles. Avec un grand nombre d'articles, une catégorisation sera nécessaire pour simplifier la recherche, et les utilisateurs pourront accéder aux articles par ancienneté, nouveauté ou popularité. Un back-office sera mis en place pour gérer les utilisateurs et les articles. La sécurité sera une priorité avec un système d'inscription et de connexion sécurisé.

# User Stories

## Avec 3 types d'utilisateur

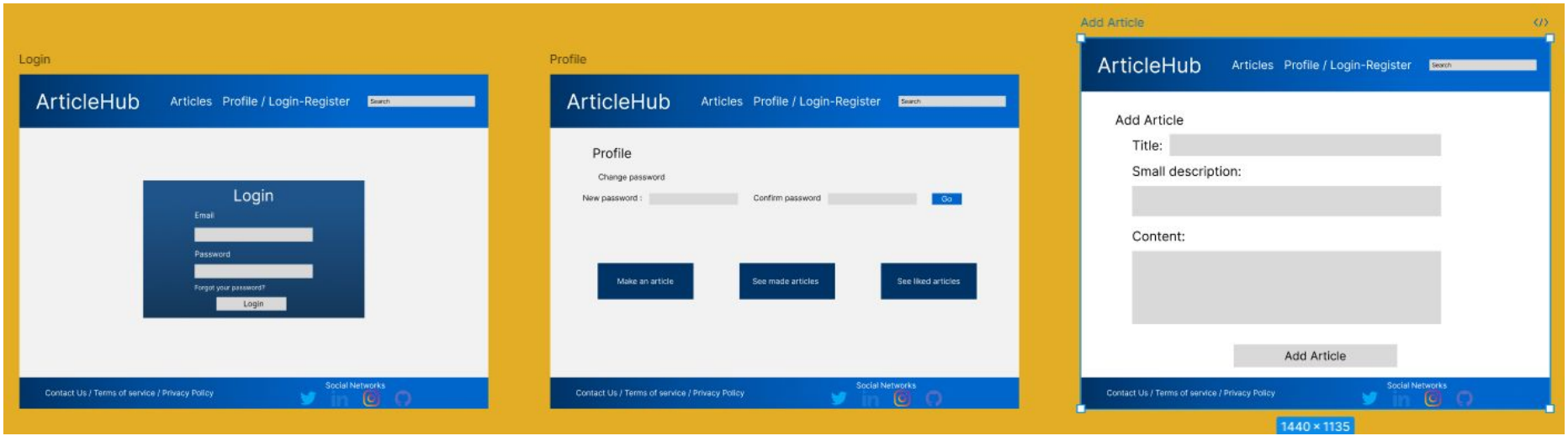
- Utilisateur non connecté
- Utilisateur connecté
- Administrateur

| ID | En tant que | Je veux | Afin de |
|---|---|---|---|
| 1 | Utilisateur non connecté | M'inscrire | Créer un compte et accéder à des fonctionnalités réservées |
| 2 | Utilisateur non connecté | Me connecter | Accéder à mon compte et interagir avec le contenu |
| 3 | Utilisateur connecté | Me déconnecter | Quitter mon compte en toute sécurité |
| 4 | Utilisateur non connecté | Consulter les articles | Lire le contenu disponible sur le site |
| 5 | Utilisateur connecté | Consulter les articles | Lire le contenu disponible et y réagir |
| 6 | Utilisateur connecté | Commenter un article | Partager mes opinions et participer aux discussions |
| 7 | Utilisateur connecté | Liker un article | Montrer mon appréciation pour le contenu |
| 8 | Utilisateur connecté | Créer un nouvel article | Partager mes connaissances ou mes opinions |
| 9 | Utilisateur connecté | Attribuer une catégorie à un article | Classer l'article pour le retrouver plus facilement |
| 10 | Utilisateur connecté | Filtrer les articles par catégorie | Trouver des articles sur des sujets spécifiques |
| 11 | Administrateur | Modérer les commentaires | Maintenir un environnement de discussion respectueux |
| 12 | Administrateur | Supprimer des articles inappropriés | Assurer la qualité et la pertinence du contenu publié |
| 13 | Administrateur | Créer et gérer des catégories | Organiser les articles de manière structurée |

# Maquette

Réalisé avec Figma

# Plus de frame équivalent à des page dans Figma

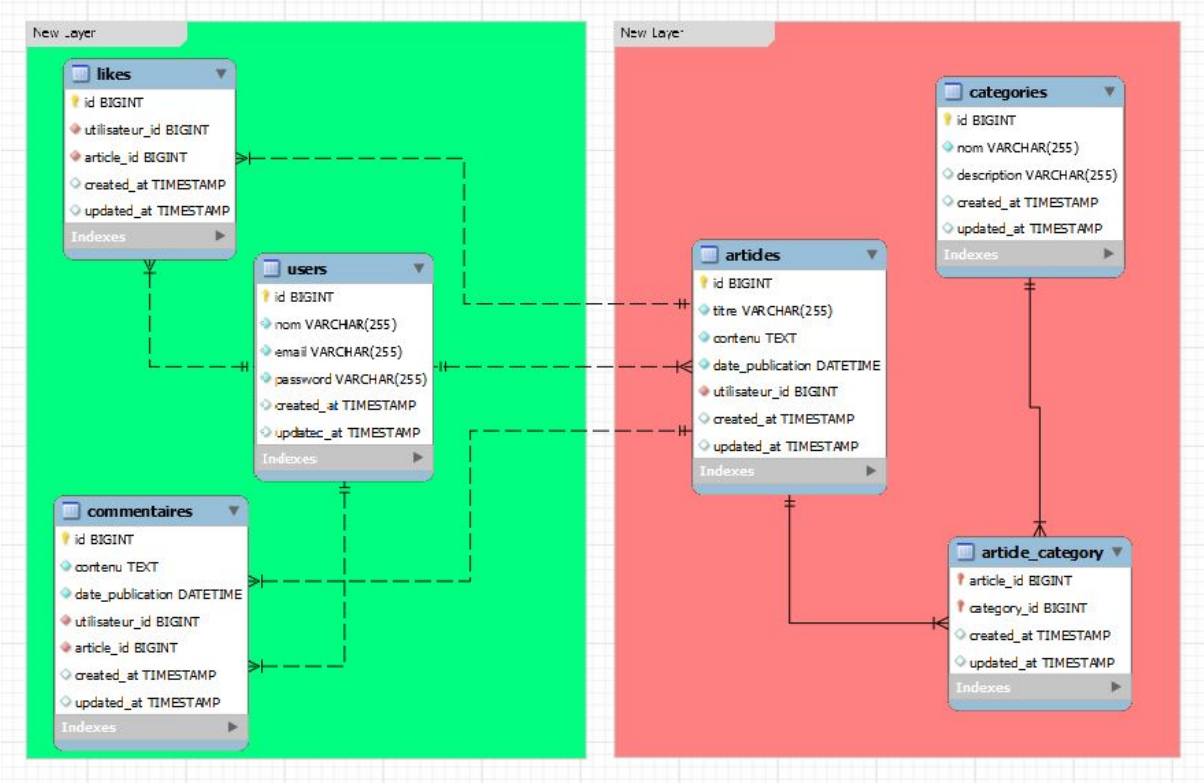


# Wireframes

Pour les WireFrames je utilise la function prototype de figma.

Voici quelque lien d'interaction entre les page:

# Conception du MCD avec mySQl workbench

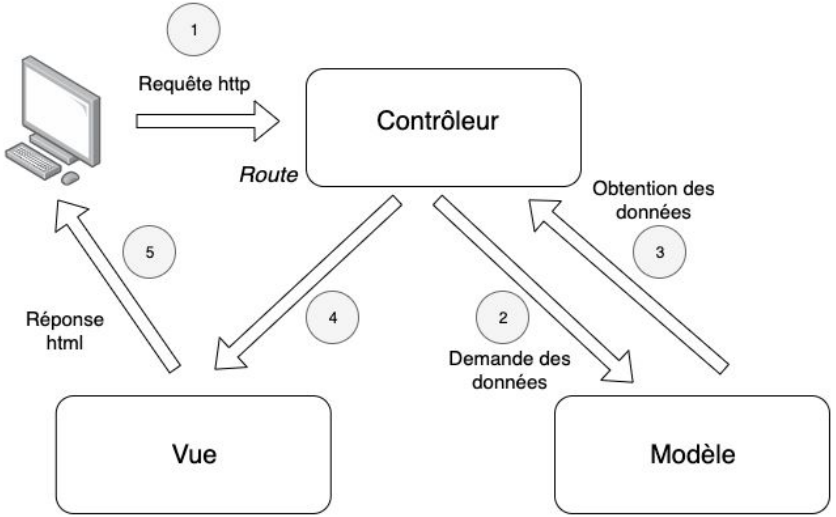


# Stack utilisé pour mon projet

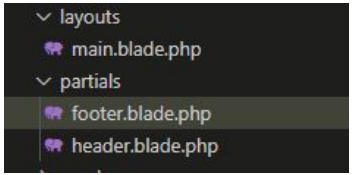
- **Frontend** :
  - HTML, CSS, JavaScript : Pour la structure, le style et l'interactivité des pages.
- **Backend** :
  - Laravel : Framework PHP pour la gestion de la logique métier et des interactions avec la base de données.
- **Base de Données** :
  - MySQL : Pour le stockage des données utilisateurs, articles, catégories, commentaires et likes.
- **Outils et Bibliothèques Supplémentaires** :
  - Blade : Moteur de template Laravel pour la génération dynamique des pages.

# Architecture

Avec le framework je suivi la architecture MVC



# Front-End et templates



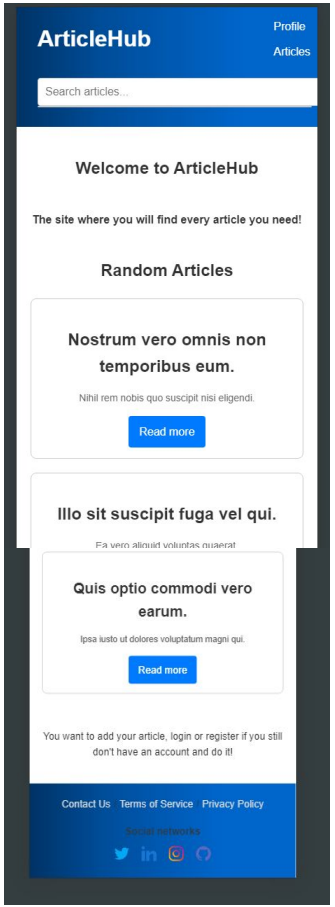

```blade
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@yield('title')</title>
    <link rel="stylesheet" href="{{ asset('css/app.css') }}">
    <link rel="stylesheet" href="{{ asset('css/back-office.css') }}">
    <link rel="stylesheet" href="{{ asset('css/auth.css') }}">
    <link rel="stylesheet" href="{{ asset('css/profile.css') }}">
    <link rel="stylesheet" href="{{ asset('css/footer.css') }}">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <script>
        window.likeFilledUrl = "{{ asset('images/like-filled.png') }}";
        window.likeUrl = "{{ asset('images/like.png') }}";
    </script>
</head>
<body>
    <!-- Header -->
    @include('partials.header')

    <!-- Main Content -->
    <div class="content">
        @yield('content')
    </div>

    <!-- Footer -->
    @include('partials.footer')

    <!-- Scripts -->
    <script src="{{ asset('js/app.js') }}"></script>
    @stack('scripts')
</body>
</html>
```

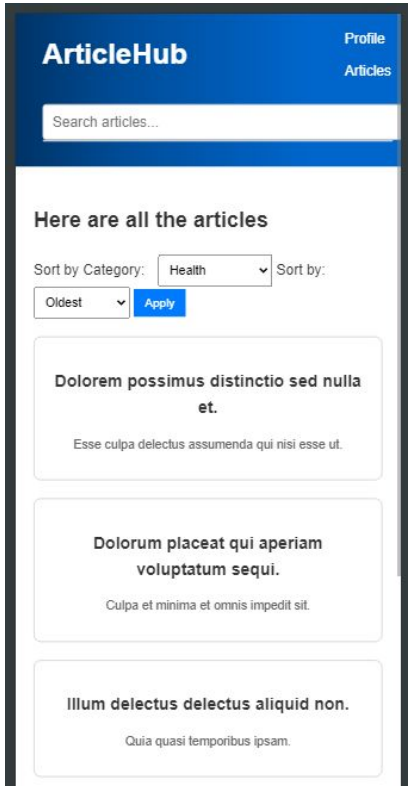
```blade
<header>
    <div class="header-content">
        <h1><a href="/" class="header-h1">ArticleHub</a></h1>
        <nav>
            @if (session('userId'))
                <a href="/profile">Profile</a>
            @else
                <a href="/auth/create">Registration</a>
                <a href="/login">Login</a>
            @endif
            <a href="/articles">Articles</a>
        </nav>
    </div>
    <div class="search-container">
        <input type="text" id="search" placeholder="Search articles..." onkeyup="searchArticles()">
        <ul id="search-results"></ul>
    </div>
</header>
```

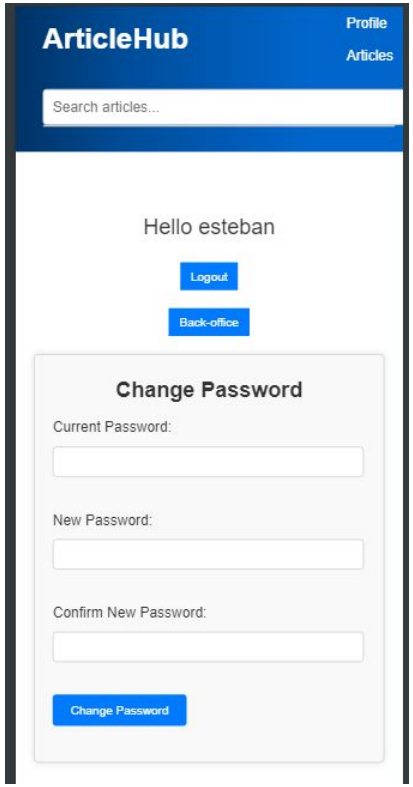
```blade
<footer>
    <div class="footer-links">
        <a href="{{ route('home')}}">Contact Us</a> /
        <a href="{{ route('home')}}">Terms of Service</a> /
        <a href="{{ route('home')}}">Privacy Policy</a>
    </div>

    <div class="footer-social">
        <h4>Social networks</h4>
        <div class="social-icons">
            <a href="https://x.com/estebanx55" target="_blank"><img
            src="{{ asset('images/twitter.png') }}" alt="Twitter"></a>
            <a href="https://www.linkedin.com/in/esteban-bare-337927284/" target="_blank"><img
            src="{{ asset('images/linkedin.png') }}" alt="Linkedin"></a>
            <a href="https://www.instagram.com/esteban5.bare/" target="_blank"><img
            src="{{ asset('images/instagram.png') }}" alt="Instagram"></a>
            <a href="https://github.com/Esteban-Bare" target="_blank"><img
            src="{{ asset('images/github.png') }}" alt="Github"></a>
        </div>
    </div>
</footer>
```

```blade
@extends('layouts.main')

@section('title', 'Liked Articles')

@section('content')
    <div class="liked-articles-container">
        <h2>Liked Articles</h2>
        @foreach ($likedArticles as $article)
        @endforeach
    </div>
@endsection
```

# Responsive




```css
@media (min-width: 600px) {
    .article-header {
        flex-direction: row;
        justify-content: space-between;
        align-items: center;
    }
    .article-meta {
        text-align: right;
    }
    .article-meta p {
        margin-left: 20px;
    }
}
@media (max-width: 599px) {
    .article-container {
        padding: 15px;
        width: 95%;
    }
    .like-section p {
        font-size: 0.9em;
    }
    .comment h4 {
        font-size: 0.9em;
    }
    header a {
        display: block;
        margin: 5px 0;
        text-decoration: none;
    }
    nav {
        flex-direction: column;
    }
    nav a {
        margin: 5px 0;
    }
}
```

```css
@media (max-width: 1200px) {
    .ah-article-grid {
        grid-template-columns: repeat(4, 1fr);
    }
}
@media (max-width: 992px) {
    .ah-article-grid {
        grid-template-columns: repeat(3, 1fr);
    }
}
@media (max-width: 768px) {
    .ah-article-grid {
        grid-template-columns: repeat(2, 1fr);
    }
}
@media (max-width: 576px) {
    .ah-article-grid {
        grid-template-columns: 1fr;
    }
}
```

# Création de la base de données

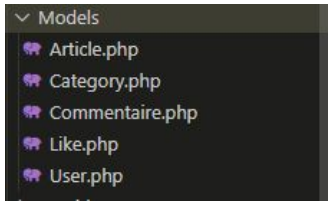
```php
class CreateUsersTable extends Migration
{
    0 references | 0 overrides
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('nom');
            $table->string('email')->unique();
            $table->string('password');
            $table->enum('role', ['user', 'admin'])->default('user');
            $table->timestamps();
        });
    }

    0 references | 0 overrides
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

```php
class CreateArticlesTable extends Migration
{
    0 references | 0 overrides
    public function up()
    {
        Schema::create('articles', function (Blueprint $table) {
            $table->id();
            $table->string('titre');
            $table->text('contenu');
            $table->text('small_description')->nullable();
            $table->dateTime('date_publication')->default(DB::raw('CURRENT_TIMESTAMP'));
            $table->foreignId('utilisateur_id')->constrained('users')->onDelete('cascade');
            $table->timestamps();
        });
    }

    0 references | 0 overrides
    public function down()
    {
        Schema::dropIfExists('articles');
    }
}
```

```php
class CreateCategoriesTable extends Migration
{
    0 references | 0 overrides
    public function up()
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->string('nom');
            $table->string('description')->nullable();
            $table->timestamps();
        });
    }

    0 references | 0 overrides
    public function down()
    {
        Schema::dropIfExists('categories');
    }
}
```

Pour créer ma base de données j'utilise les migrations de Laravel qui m'ont permis de créer c'est table de maniere tres simple avec la command 'php artisan migrate'

# Composant d'accès aux données




```php
class Article extends Model
{
    use HasFactory;

    0 references
    protected $fillable = ['titre', 'contenu', 'date_publication', 'utilisateur_id', 'small_description'];

    0 references | 0 overrides
    public function user()
    {
        return $this->belongsTo(User::class, 'utilisateur_id');
    }

    0 references | 0 overrides
    public function commentaires()
    {
        return $this->hasMany(Commentaire::class, 'article_id');
    }

    4 references | 0 overrides
    public function likes()
    {
        return $this->hasMany(Like::class, 'article_id');
    }

    0 references | 0 overrides
    public function userHasLiked($userId)
    {
        return $this->likes()->where('utilisateur_id', $userId)->exists();
    }

    2 references | 0 overrides
    public function categories()
    {
        return $this->belongsToMany(Category::class, 'article_category', 'article_id', 'category_id');
    }
}
```

```php
class User extends Authenticatable
{
    use HasFactory, Notifiable;
    0 references
    protected $fillable = [
        'nom',
        'email',
        'password',
        'role', // Add the role field here
    ];
    1 reference | 0 overrides
    public function createdArticles()
    {
        return $this->hasMany(Article::class, 'utilisateur_id');
    }
    0 references | 0 overrides
    public function commentaires()
    {
        return $this->hasMany(Commentaire::class, 'utilisateur_id');
    }
    1 reference | 0 overrides
    public function likes()
    {
        return $this->hasMany(Like::class, 'utilisateur_id');
    }
    0 references | 0 overrides
    public function isAdmin()
    {
        return $this->role === 'admin';
    }
    0 references | 0 overrides
    public function isUser()
    {
        return $this->role === 'user';
    }
}
```

J'ai utilisé les modèles créés par Laravel après la migration des tables vers la base de données. Par exemple, j'ai utilisé les modèles User et Article, ainsi que leurs relations.

# Routes

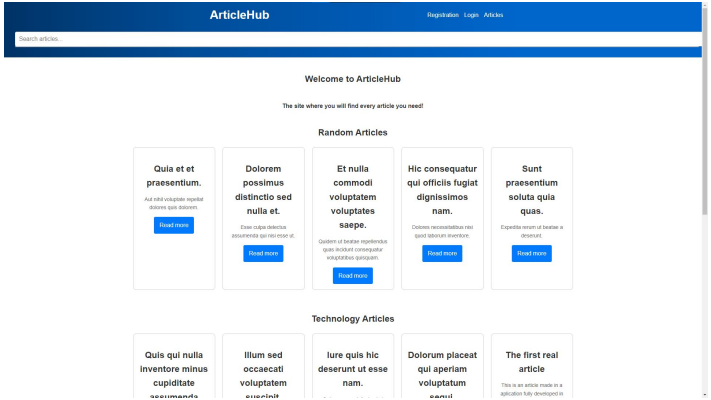
| Endpoint | Méthode | Fonctionnalité | Contrôleur |
|---|---|---|---|
| / | GET | Afficher la page d'accueil | HomeController |
| /profile | GET | Afficher le profil de l'utilisateur | ProfileController |
| /profile/liked | GET | Afficher les articles aimés par l'utilisateur | ProfileController |
| /profile/articles | GET | Afficher les articles de l'utilisateur | ProfileController |
| /profile/articles/{id} | DELETE | Supprimer un article de l'utilisateur | ProfileController |
| /profile/password/change | POST | Changer le mot de passe de l'utilisateur | ProfileController |
| /admin | GET | Afficher la vue d'administration | AdminController |
| /admin/users | GET | Afficher la liste des utilisateurs | AdminController |
| /admin/articles | GET | Afficher la liste des articles | AdminController |
| /admin/articles/{id} | DELETE | Supprimer un article | AdminController |
| /admin/users/{id} | DELETE | Supprimer un utilisateur | AdminController |
| /auth | Resource | Gérer l'authentification (CRUD) | RegistrationController |
| /login | GET | Afficher le formulaire de connexion | AuthController |
| /login | POST | Gérer la connexion utilisateur | AuthController |
| /logout | POST | Déconnexion de l'utilisateur | AuthController |

| | | | |
|---|---|---|---|
| /articles | GET | Afficher tous les articles | ArticlesController |
| /articles/category/{category} | GET | Afficher les articles d'une catégorie spécifique | ArticleController |
| /articles/create | GET | Afficher le formulaire de création d'article | ArticlesController |
| /articles/create | POST | Créer un nouvel article | ArticlesController |
| /article/{id} | GET | Afficher un article spécifique | ArticleController |
| /article/{id}/like | POST | Gérer la fonction "J'aime" d'un article | ArticleController |
| /article/{id}/comments | POST | Créer un commentaire sur un article | CommentController |
| /search | GET | Effectuer une recherche d'articles | ArticleController |

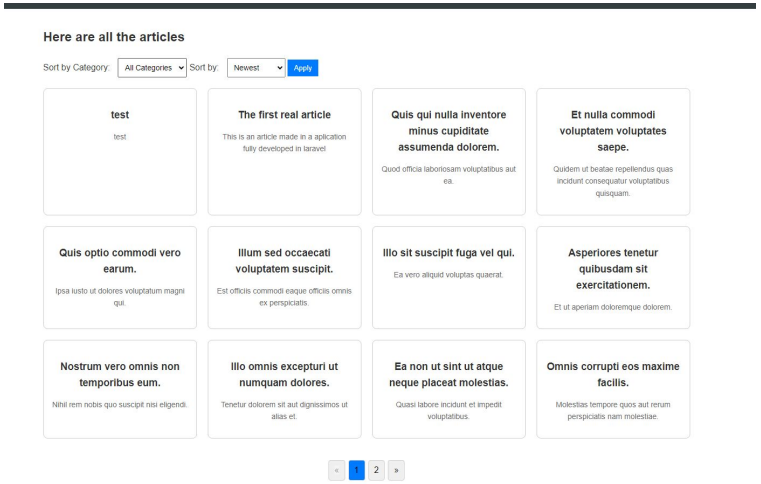
# Key Features et Fonctionnalité

# Page d'accueil et page d'articles

```php
Route::get('/', [HomeController::class, 'viewHome'])->name('home');
```



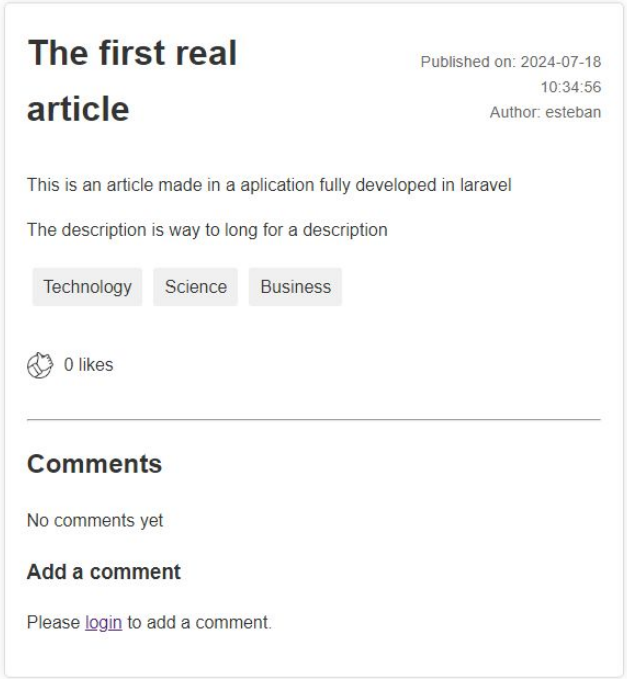

```php
Route::get('articles' , [ArticlesController::class, 'showAllArticlesView'])->name('articles.index');
```




```blade
<div class="ah-article-card">
    <h3 class="ah-article-title">{{ $article->titre }}</h3>
    <p class="ah-article-description">{{ $article->small_description }}</p>
    <a href="{{ route('article.show', ['id' => $article->id]) }}" class="ah-read-more-link">Read more</a>
</div>
```

```blade
    <a href="{{ route('article.show', ['id' => $article->id]) }}">
        <div class="article-content">
            <h3>{{ $article->titre }}</h3>
            <p>{{ $article->small_description }}</p>
        </div>
    </a>
```

# Lecture d'article

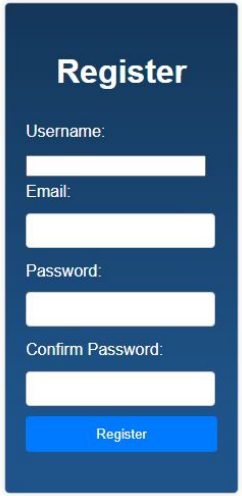



```php
Route::get('article/{id}', [ArticleController::class, 'showArticle'])->name('article.show');
```

```php
1 reference | 0 overrides
public function showArticle($id){
    $article = Article::find($id);

    return view('articles.show', ['article' => $article]);
}
```

```blade
@extends('layouts.main')

@section('title', $article->titre)

@section('content')
<div class="article-container">
    <div class="article-header">
        <h1>{{ $article->titre }}</h1>
        <div class="article-meta">
            <p>Published on: {{ $article->date_publication }}</p>
            <p>Author: {{ $article->user->nom }}</p>
        </div>
    </div>

    <p>{{ $article->small_description }}</p>
    <p>{{ $article->contenu }}</p>

    <!-- Display categories -->
    <ul>
        @foreach ($article->categories as $category)
            <li>{{ $category->nom }}</li>
        @endforeach
    </ul>

    <div class="like-section">
        <img src="{{ $article->userHasLiked(session('userId')) ? asset('images/like-filled.png') : asset('images/like.png') }}"
            alt="Like" class="like-button" data-article-id="{{ $article->id }}">
        <p>{{ $article->likes->count() }} {{ Str::plural('like', $article->likes->count()) }}</p>
    </div>

    <hr>

    <div class="comments-section">...
    </div>
</div>
@endsection
```

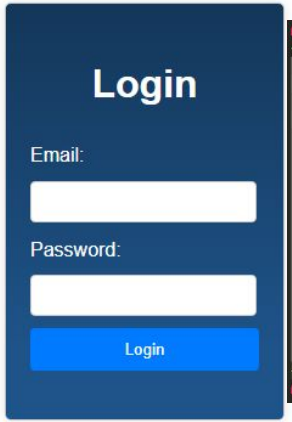
# Enregistrement et connexion




```blade
@section('content')
<div class="login-form">
    <h1 class="form-heading-create">Register</h1>
    @if (session('success'))···
    @endif
    @if ($errors->any())···
    @endif
    <form action="{{ route('auth.store') }}" method="POST" class="form-create">
        @csrf
        <div class="form-group-create">
            <label for="username" class="form-label-create">Username:</label>
            <input type="text" id="username" name="username" class="form-control-create" required>
        </div>
        <div class="form-group-create">
            <label for="email" class="form-label-create">Email:</label>
            <input type="email" id="email" name="email" class="form-control-create" required>
        </div>
        <div class="form-group-create">
            <label for="password" class="form-label-create">Password:</label>
            <input type="password" id="password" name="password" class="form-control-create" required>
        </div>
        <div class="form-group-create">
            <label for="password_confirmation" class="form-label-create">Confirm Password:</label>
            <input type="password" id="password_confirmation" name="password_confirmation" class="form-control-create" required>
        </div>
        <button type="submit" class="btn btn-primary btn-register-create">Register</button>
    </form>
</div>

@endsection
```

```php
Route::resource('auth', RegistrationController::class);
```

```php
public function store(Request $request)
{
    $request->validate([
        'nom' => 'required|string|max:255',
        'email' => 'required|string|email|max:255|unique:users',
        'password' => 'required|string|min:8|confirmed',
    ]);

    $user = User::create([
        'nom' => $request->input('nom'),
        'email' => $request->input('email'),
        'password' => bcrypt($request->input('password')),
    ]);

    return redirect()->route('auth.login')->with('success', 'Registration successful!');
}
```

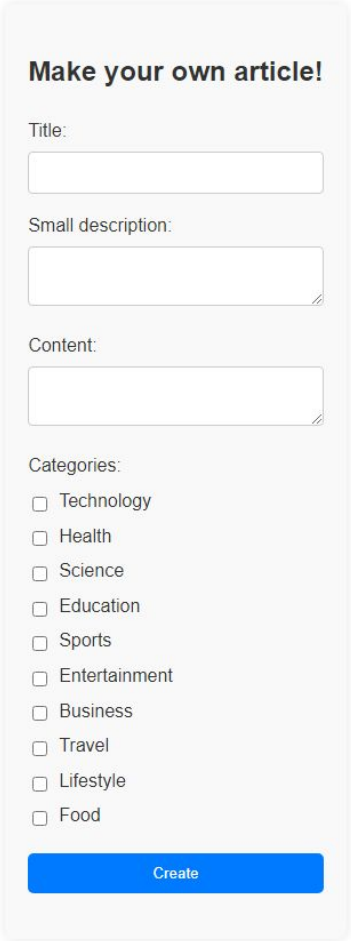



```blade
@section('content')
<div class="login-form">
    <h1 class="form-heading-login">Login</h1>
    @if (session('success'))···
    @endif
    @if ($errors->any())···
    @endif
    <form action="{{ route('login') }}" method="POST" class="form-login">
        @csrf
        <div class="form-group-login">
            <label for="email" class="form-label-login">Email:</label>
            <input type="email" id="email" name="email" class="form-control-login" value="{{ old('email') }}" required>
        </div>
        <div class="form-group-login">
            <label for="password" class="form-label-login">Password:</label>
            <input type="password" id="password" name="password" class="form-control-login" required>
        </div>
        <button type="submit" class="btn btn-primary btn-login">Login</button>
    </form>
</div>
@endsection
```

```php
Route::post('login', [AuthController::class, 'login'])->name('login');
```

```php
public function login(Request $request) {
    $credentials = $request->only('email','password');

    if(Auth::attempt($credentials)) {
        $request->session()->put('userId', Auth::user()->id);
        $request->session()->put('userEmail', Auth::user()->email);

        return redirect()->intended('/');
    }

    return back()->withInput()->withErrors(['email'=> 'These credentials do not match our records.']);
}
```

# Création d'article

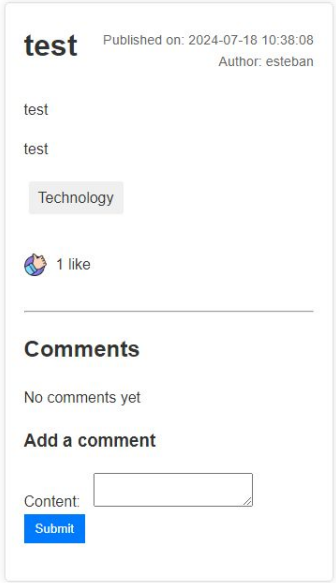



```php
Route::get('articles/create' , [ArticlesController::class, 'showCreateArticleView'])->name('articleCreate');

Route::post('articles/create', [ArticlesController::class , 'createArticle'])->name('createArticle');
```

```php
public function showCreateArticleView() {
    $categories = Category::all();

    return view('articles.createArticle')->with('categories', $categories);
}

1 reference | 0 overrides
public function createArticle(Request $request) {

    $request->validate([
        'titre' => 'required|unique:articles,titre',
        'contenu' => 'required',
        'small_description' => 'nullable|string|max:255',
        'categories' => 'required|array',
        'categories,*' => 'exists:categories,id',
        'utilisateur_id' => 'required|exists:users,id',
    ]);

    $article = Article::create($request->only(['titre','contenu','small_description','utilisateur_id', 'date_publication']));

    $article->categories()->attach($request->categories);

    return redirect()->route('articles.index')->with('success', 'Article created successfully.');
}
```

# Fonctionnalité “Like”

```php
Route::post('/article/{id}/like', [ArticleController::class, 'like'])->name('article.like');
```



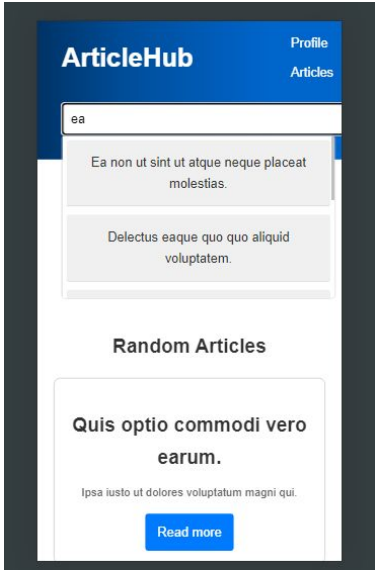

```blade
<div class="like-section">
    <img src="{{ $article->userHasLiked(session('userId')) ? asset('images/like-filled.png') : asset('images/like.png') }}"
        alt="Like" class="like-button" data-article-id="{{ $article->id }}">
    <p>{{ $article->likes->count() }} {{ Str::plural('like', $article->likes->count()) }}</p>
</div>
```

```php
public function likes()
{
    return $this->hasMany(Like::class, 'article_id');
}

0 references | 0 overrides
public function userHasLiked($userId)
{
    return $this->likes()->where('utilisateur_id', $userId)->exists();
}
```

```php
public function like($id)
{
    $article = Article::findOrFail($id);
    $userId = session('userId');

    if ($userId) {
        if ($article->likes()->where('utilisateur_id', $userId)->exists()) {

        $article->likes()->where('utilisateur_id', $userId)->delete();
        return response()->json(['status' => 'unliked']);
    } else {
        $article->likes()->create(['utilisateur_id' => $userId]);
        return response()->json(['status' => 'liked']);
    }
    } else {
        return response()->json(['status' => 'unliked']);
    }


}
```

```javascript
document.querySelectorAll('.like-button').forEach(button => {
    button.addEventListener('click', function() {
        const articleId = this.dataset.articleId;
        if (!articleId) {
            console.error('Article ID not found');
            return;
        }

        fetch(`/article/${articleId}/like`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'X-CSRF-TOKEN': token
            },
            body: JSON.stringify({})
        })
        .then(response => response.json())
        .then(data => {
            if (data.status === 'liked') {
                // console.log('liked');
                this.src = window.likeFilledUrl;
            } else {
                // console.log('unliked');
                this.src = window.likeUrl;
            }
        })
        .catch(error => console.error('Error:', error));
    });
});
```

# Barre de recherche



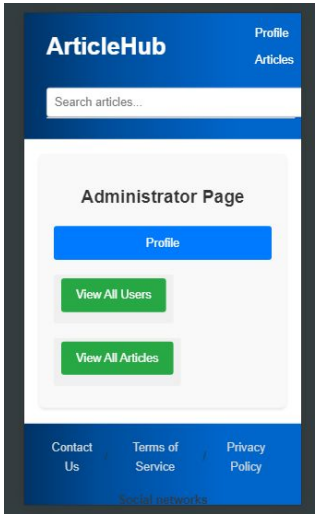

```php
Route::get('/search', [ArticleController::class, 'search'])->name('articles.search');
```

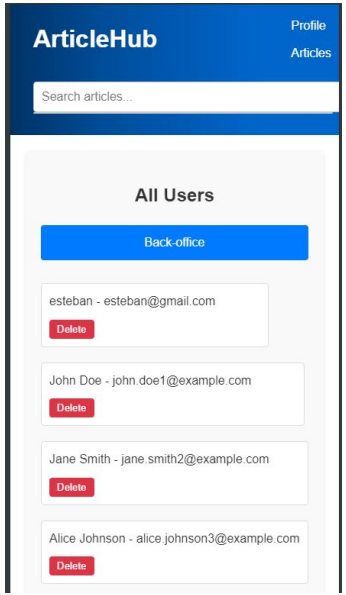
```html
<div class="search-container">
    <input type="text" id="search" placeholder="Search articles..." onkeyup="searchArticles()">
    <ul id="search-results"></ul>
</div>
```

```javascript
async function searchArticles() {
        const query = document.getElementById('search').value;
        const searchResults = document.getElementById('search-results');
        searchResults.innerHTML = '';

        if (query.length < 2) {
            return;
        }

        const response = await fetch(`/search?query=${query}`);
        const articles = await response.json();

        articles.forEach(article => {
            const li = document.createElement('li');
            li.innerHTML = `<a href="/article/${article.id}">${article.titre}</a>`;
            searchResults.appendChild(li);
        });
    }
```

```php
public function search(Request $request)
{
    $query = $request->input('query');

    if ($query) {
        $articles = Article::where('titre', 'LIKE', "%{$query}%")->get();
        return response()->json($articles);
    }

    return response()->json([]);
}
```

# Back-Office




```php
Route::get('/admin', [AdminController::class, 'showAdminView'])->name('admin.view');
Route::get('/admin/users', [AdminController::class, 'showUsers'])->name('admin.users');
Route::get('/admin/articles', [AdminController::class, 'showArticles'])->name('admin.articles');
Route::delete('/admin/articles/{id}', [AdminController::class, 'deleteArticle'])->name('admin.deleteArticle');
Route::delete('/admin/users/{id}', [AdminController::class, 'deleteUser'])->name('admin.deleteUser');
```

```php
public function showAdminView() {
    $userId = session('userId');

    if (!$userId) {
        return redirect()->route('auth.login')->with('error', 'You must be logged in to view this page.');
    }
                    (local variable) mixed|SessionManager|Store $userId
    $user = User::find($userId);

    if (!$user) {
        return redirect()->route('home')->with('error', 'User not found.');
    }

    if ($user->role === "admin") {
        return redirect()->route('home')->with('error', 'User not found.');
    }

    return view('admin.index');
}
```

```blade
@foreach($users as $user)
    <li>
        <div>{{ $user->nom }} - {{ $user->email }}</div>
        <form action="{{ route('admin.deleteUser', $user->id) }}" method="POST" style="display:inline">
            @csrf
            @method('DELETE')
            <button type="submit">Delete</button>
        </form>
    </li>
@endforeach
```

```php
public function deleteUser($id) {
    $user = User::find($id);
    if ($user) {
        $user->delete();
    }
    return redirect()->route('admin.users')->with('success', 'User deleted successfully');
}
```

# Déploiement

“git clone https://github.com/Esteban-Bare/Project-ArticleHub”

“cd votre-projet”

“composer install”

.env.example -> .env

“php artisan key:generate”

“php artisan migrate”

“php artisan db:seed”

“php artisan serve”

# Conclusion

- Je suis satisfait de mon projet accompli.
- Content de mon apprentissage de Laravel.
- Envie d'aller plus loin.

Merci d'avoir écouté.

Des questions ?