Portfolio Project - Project Closure

Lord of the Ring – Interactive Wiki

1. RESULTS SUMMARY

1.1 Project overview

Total duration : 13 August - 31 October 2024 (two and a half months)

- Documentation phase : 13 August - 28 September
- Development phase: 29 September - 31 October

Team : 2 ddevelopers

1.2 Main features of the MVP

The delivered MVP includes the following features:

☑ Implemented features (90% of the MVP)

Interactive Map 

- Interactive Middle-earth map
- Clickable marker system with informative pop-ups
- Leaflet integration with React

Authentication System 

- User account creation
- Secure login/logout
- Session management with JWT
- Password hashing with bcrypt
- Guest mode with content restrictions

Dynamic wiki 

- 13 main functional pages
- Architecture based on reusable React components
- Fully dynamic content (races, characters, stories)
- All data stored and retrieved from PostgreSQL

Art gallery 

- Image uploads for logged-in users
- Comment system for each creation
- Gallery display and navigation

User experience 

- Design appreciated by test users
- Intuitive navigation with the addition of a navigation bar to quickly navigate to each different menu with the addition of an intuitive logo

❌ **Features not implemented (10% of initial MVP)**

- Administrator mode for modifying content via interface
- Advanced animation effects on the map
- Filter system by content type
- Fully responsive interactive map on mobile devices

1.3 Comparison with initial objectives

Initial objective	Status	Comment
Interactive Lord of the Rings map	Validated	Functional with markers added in the right places and pop-ups appearing when the user clicks.
Artwork gallery	Validated	All data is stored in a database, 13 pages created via React.
Galerie de créations artistiques	Validated	Image uploading and comments on images are functional.
Authentication system	Validated	Secure with JWT and bcrypt
Guest mode	Validated	Restricted access implemented for the gallery section and no possibility to modify informational texts
Admin mode	Partial	Back-end management only
Mobile responsive	Partial	Completed except for the map, which was too complex
Map animations	Not validated	Technical complexity and lack of time
Filter system	Not validated	Technical complexity

1.4 Key Metrics

Technical

- 13 main functional pages
- Complete frontend architecture (React + Vite) / backend architecture (Flask)
- PostgreSQL database with complex relational tables
- 100% of API calls functional
- Complete error handling on critical features

Users feedback

- ☑ Highly appreciated design
- ☑ Intuitive navigation
- ⚠️ Mobile experience could be improved on the map
- ☑ Smooth image upload system

2. LESSONS LEARNED

2.1 What worked well

Solid technical architecture

What happened: Using Flask with its extensions (RestX, SQLAlchemy, JWT) coupled with React made it possible to create a clear and maintainable architecture.

Why it was effective:

- Clear separation of frontend/backend
- Reusable React components
- Well-structured REST API with Flask-RESTX
- SQLAlchemy ORM facilitating database management

Keep for future projects: This modern technology stack and component-based architecture.

Dynamic content management

What happened: The decision to store 100% of the content in a database rather than in static HTML.

Why it was effective:

- Facilitates content updates
- Project scalability
- Data consistency
- Future possibility of adding a true admin mode

Keep for future projects : Always prioritise dynamic content from the outset.

Secure authentication system

What happened : Full implementation with JWT, bcrypt, and session management.

Why it was effective:

- User data security
- Smooth user experience
- Robust error handling

To keep for future projects: Do not neglect security from the MVP stage onwards.

Preliminary documentation phase

What happened: Six weeks dedicated to documentation and learning before starting development.

Why it was effective :

- Clear vision of the project
- Reduced hesitation during development
- Better organisation of pair work
- Learning React and deciding which languages to use

To keep for future projects: Invest time in planning.

2.2 Challenges encountered and solutions provided

Challenge 1: Interactive map with Leaflet and React

Problem: Complex integration of Leaflet into the React ecosystem, particularly the management of markers and pop-ups.

How it was resolved:

- In-depth study of Leaflet documentation
- Use of artificial intelligence**Impact :** Carte fonctionnelle mais responsive mobile limité.

Recommandation for the future : Allow more time for mapping features

Challenge 2: Learning React in a real-world situation

Problem: React learning curve (components, props, state, hooks) during development.

How it was solved:

- Progressive training through practice
- Refactoring code as understanding progressed
- Pair programming to share knowledge
- Requesting information from other students

Impact: Some slowdowns at the beginning of development, but rapid skill improvement.

Future recommandation : Allow for a longer technical training phase before starting development.

Défi 3 : Challenge 3: Scalable database structure

Problem: The initial database schema required adjustments (tables lacking information, tables too complex, difficult polymorphic relationships).

How it was resolved :

- Simplification of certain complex relationships

Impact: Loss of time because the database was deconstructed several times and often restarted

Recommendation for the future: Keep the database design simple. For a beginner's project, the database was probably too complex.

Challenge 4 : Gestion de l'upload d'images avec JavaScript

Problem: Complexity of managing multipart/form-data files between the frontend and backend.

How it was solved:

- Use of artificial intelligence
- Exhaustive testing with different

Impact: Smooth and functional final upload system.

Future recommendation: Anticipate the complexity of file management in planning.

Problem: Complexity of managing multipart/form-data files between the frontend and backend.

How it was solved:

- Use of artificial intelligence
- Exhaustive testing with different

Impact: Smooth and functional final upload system.

Future recommendation: Anticipate the complexity of file management in planning.

Challenge 5: Polymorphic tables and complex relationships

Problem: Difficulty understanding and implementing polymorphic relationships in SQLAlchemy.

How it was solved:

- Simplification of certain relationships
- Simpler alternative schemas (tables)

Impact : Time wasted rebuilding a table and modifying models, routes, and facades.

Future recommendation: Evaluate the complexity/benefit ratio of polymorphic relationships. Sometimes a simpler solution is preferable.

2.3 Areas for improvement in future projects

1. Responsive design from the outset

Observation: Responsive design was addressed at the end of the project, but on certain pages, it would have been preferable to consider it at the outset, given that it is a prerequisite for the portfolio.

Improvement: Test on mobile devices throughout development.

2. Estimating time for advanced features

Observation: Some planned features (animations, filters) were not implemented due to lack of time and code complexity.

Improvement : We spent a lot of time developing it on the back end, only to end up not using it on the front end of our site. We should have made this type of feature optional from the outset and only developed it at the end.

3. Synchronisation of diagrams with the code

Observation: The initial diagrams no longer correspond to the final product.

Improvement: Update the technical documentation as the project progresses, particularly after each significant change to the database.

2.4 Recommendations for the rest of the project

Short term

1. Correct the last remaining issues
2. Update the technical diagrams to reflect the current status
3. Decide on the mobile strategy for the map (removal with message or map always present, it will be less manipulable than on a PC)

Medium term

1. Implement administrator mode with graphical interface
2. Implement user mode with password change or other personal information
3. Optimise image loading performance
4. Improve responsiveness on mobile devices

Long term (project development)

1. Add animations to the map
2. Advanced search system in the wiki
3. Add a filter system by content type

3. CONCLUSION

The MVP of the interactive Lord of the Rings Wiki **has achieved 90% of its initial objectives.** The core features (interactive map, dynamic wiki, authentication, art gallery) are all operational and appreciated by test users.

The main technical challenges (Leaflet-React integration, learning React, database management) were overcome thanks to the team's perseverance and hard work and an iterative approach.

This project enabled the team to develop solid skills in modern full-stack development and deliver a functional and aesthetically successful product.

Project highlights:

- Robust and scalable technical architecture
- User-friendly design
- Secure authentication system
- Comprehensive dynamic content management

Areas for improvement identified:

- Mobile responsiveness on complex components
- Time estimation for advanced features
- Earlier user testing
- Shorter page loading times (plan for pagination)