Introduction to Development

At Unicis, our development process is the backbone of creating secure and compliant solutions. This section outlines our methodology, team structure, and best practices to ensure our products meet the highest standards.

Development Philosophy and Approach

Agile, but not limited to and not strictly by the book.

Iterative, incremental, and evolutionary

Agile development methodologies divide product development into small, incremental iterations known as sprints, which typically span from one to four weeks. Each sprint involves a cross-functional team engaged in planning, analysis, design, coding, unit testing, and acceptance testing, culminating in a functional product demonstration to stakeholders. This approach minimizes initial planning, reduces overall risk, and enables the product to swiftly adapt to modifications. The goal is to make a release that is minimally buggy, so that issues can be fixed quickly. Agile methodologies have several key advantages, including a faster time to market and a reduced risk of developing products that do not meet user requirements.

Efficient and face-to-face communication

The sixth principle of the Agile Manifesto stresses the importance of face-to-face conversation as the most effective way to convey information within a development team. This idea was written before video conferencing was common, but it still suggests team co-location for better communication and team identity. During the COVID-19 pandemic, research has demonstrated that effective remote working tools make co-location less critical. Each team should have a customer representative (product owner in Scrum) to answer questions and align priorities, ensuring stakeholder satisfaction and optimizing ROI. Agile development also relies on information radiators, which are large, visible displays showing the project’s status (at Unicis we use OpenProject).

Very short feedback loop and adaptation cycle

In agile development, the daily stand-up, or daily scrum, is a brief 15-minute meeting where team members review progress, plans, and impediments. Simple questions are used to keep the meeting short, with detailed discussions postponed until afterward.

Quality focus

Agile development often employs tools and techniques like continuous integration, automated testing, pair programming, and test-driven development to improve quality and agility. These practices focus on building quality from the start and demonstrating software to customers regularly, at least by the end of each iteration.

Development Team Structure

Role	Responsibilities
Developers	Agile developers are in charge of writing and testing code, working with team members, adapting to changes, maintaining documentation, and improving processes and skills.
Designers	Designers in agile development create user interfaces, make sure everything works well for users, work with developers and other people involved, get feedback from users, and change designs throughout the process.
DevOps	DevOps responsibilities include managing the software development lifecycle, facilitating continuous integration and continuous deployment (CI/CD), ensuring system reliability and performance, working with development and operations teams, monitoring system health, and managing infrastructure and configuration.
Test / QA	The job of a test/QA person is to make plans for testing, find and report bugs, make sure the software works well, work with other developers, check fixes, and keep getting better at testing.

Development Process

Project Planning

A process that changes constantly and adapts to customer needs and business goals.

Roadmapping

Aligns long-term goals with development milestones.

Sprint Planning

Detailed planning of tasks for each sprint, ensuring alignment with project goals.

Code Review and Quality Assurance

Work together to check the code to make sure it follows the rules, finds problems early, and keeps the software good all the time.

Code Review

Peer reviews to maintain high code quality.

Testing

Combination of automated and manual testing to ensure functionality, performance, and security.

Continuous Integration/Continuous Deployment (CI/CD)

Methodology that makes it easier to integrate, test, and deploy code changes.

CI/CD Pipeline

Automated testing and deployment processes to ensure quick and reliable delivery.

Tools and Technologies

Programming Languages and Frameworks

Our stack includes Development Tools: Version control, project management, team communication, IDE's.

CI/CD Tools: GitLab CI (primary), GitHub Actions (Community Edition), CapRover (deployment). See DevOps for full pipeline documentation.

Documentation

Importance of Documentation
Ensures clarity, facilitates onboarding, and maintains consistency.

Types of Documentation

Our organization has a lot of different needs and audiences. Internally, we maintain comprehensive technical documentation that is specifically tailored for developers, ensuring clarity and guidance throughout the development process. For end-users, we offer user-friendly guides and manuals that simplify product understanding and usage. Our API documentation is comprehensive, ensuring seamless integration with third-party developers. To ensure quality, we prioritize regular updates and thorough reviews, ensuring all documentation remains current, accurate, and valuable to our stakeholders and users alike.

Best Practices and Standards

Coding Standards

Adhering to consistent coding practices is fundamental in maintaining code quality and readability across our development teams. By following established guidelines and conventions, we ensure that our codebase is cohesive, maintainable, and easy to collaborate on.

Security Practices

Security is paramount in safeguarding sensitive data and ensuring compliance with industry regulations. Implementing robust security measures, such as encryption protocols, access controls, and regular security

audits, helps mitigate risks and protect our systems from potential vulnerabilities.

Performance Optimization

Optimizing performance involves employing techniques like code refactoring, caching strategies, and efficient algorithms to enhance system responsiveness and resource utilization. By continually monitoring and optimizing our applications, we strive to deliver optimal user experiences and reduce operational costs.

Scalability Considerations

Ensuring scalability involves designing our systems to handle increasing workloads and user demands without compromising performance or reliability. Through scalable architecture patterns, load testing, and cloud infrastructure management, we prepare our products to grow seamlessly as our user base expands.

Additional Resources

For further best practices, we adhere to the principles outlined in the 12 Factor App methodology which guides our approach to building modern, scalable applications. Moreover, we implement the principles of the Minimum Viable Secure Product (MVSP) framework to prioritize essential security measures from the earliest stages of product development.

Training and Development

Onboarding

We prioritize comprehensive processes for integrating new developers into our team, ensuring they receive the necessary resources, training, and introductions to our workflows and tools.

Ongoing Training

Continued education and skill enhancement are facilitated through ongoing training programs that cover emerging technologies, industry best practices, and specialized areas relevant to our team’s goals and projects.

Knowledge Sharing

We foster a collaborative environment by actively encouraging information exchange among team members. This includes regular meetings, workshops, and platforms for sharing insights, experiences, and solutions.

Feedback and Improvement

Gathering Feedback

To maintain alignment with user expectations and stakeholder needs, we employ robust mechanisms for collecting and analyzing feedback through surveys, usability tests, and direct interactions with users.

Incorporating Feedback

Feedback loops are integral to our development process, enabling us to iteratively update and improve our products. We prioritize actioning feedback promptly to address issues, refine features, and enhance overall user satisfaction.

Continuous Improvement

We are committed to ongoing refinement of our processes and products. Through retrospectives, metrics analysis, and proactive problem-solving, we strive to achieve continuous improvement in efficiency, quality, and customer value.

Future Trends and Innovations

Industry Trends

Staying ahead of industry trends and regulatory changes is essential. We invest in monitoring and analyzing market shifts, technological advancements, and evolving user expectations to anticipate and adapt to future demands.

New Technologies

Exploration and adoption of innovative tools and frameworks are actively encouraged to enhance our technical capabilities and maintain competitiveness in the market. We evaluate new technologies through proofs of concept and pilot projects.

Encouraging Innovation

We nurture a culture of innovation by empowering team members to propose and experiment with new ideas. This includes dedicated time for research and development, cross-functional collaboration, and recognition of innovative contributions.

agile, development, best practice, framework, training, CI/CD, sprint, planing, DevOps, code review, testing

From:
https://handbook.unicis.tech/ - **Unicis Handbook**

Permanent link:
https://handbook.unicis.tech/pub/development?rev=1780843487

Last update: **07.06.2026 14:44**