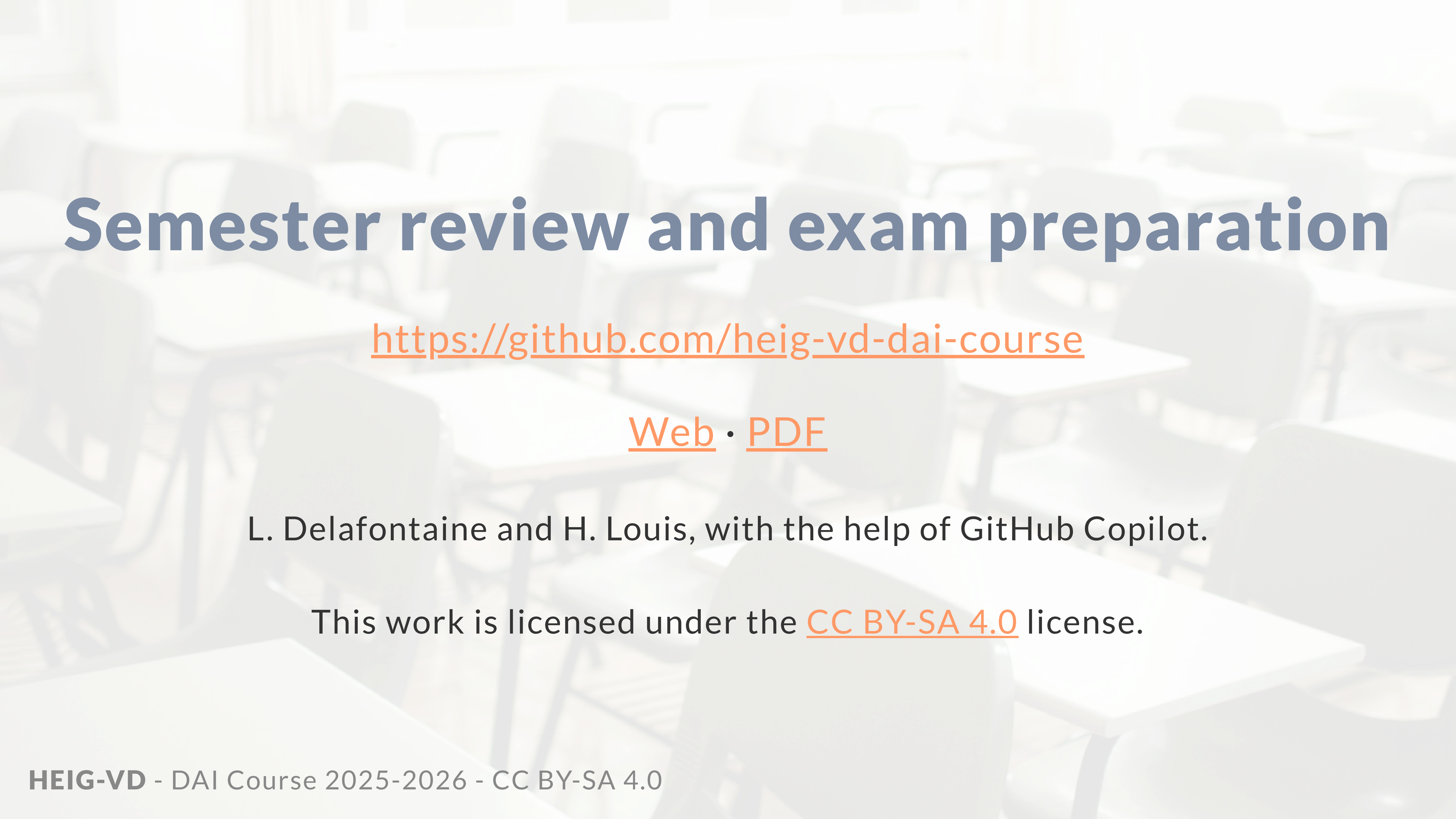

Semester review and exam preparation

https://github.com/heig-vd-dai-course

Web · PDF

L. Delafontaine and H. Louis, with the help of GitHub Copilot.

Semester review

You made it! Congratulations! 🎉

Retrospective

Let's have a look back on what **you** did during this semester.

> *You will learn the following topics during this course:*
>
> - *Network programming (inputs/outputs, encodings, TCP and UDP)*
> - *Application-level protocols (SMTP, SSH, HTTP and your own)*
> - *Web infrastructures (reverse proxy, load balancer, sticky sessions)*
> - *How to read and write technical documentation*
>
> *At the end of the course, you will be able to create applications that can communicate over the network!*

> *You will learn the following technologies during this course:*
>
> - *Git and GitHub*
> - *Markdown*
> - *Java for network programming*
> - *Docker & Docker Compose*
> - *Network utilities*
> - *The terminal* ❤️

Introduction and course organization

Set up a Windows development environment

+

Considerations for a development environment

[Find this chapter on GitHub]

This chapter will not be in the exam!

Key points to remember for this chapter

- Setting up a professional development environment can really help you be more efficient
- You have gained valuable Linux experience using WSL
- For us, the usage of WSL was really positive:
  - Less bugs/edge-cases
  - Easy access to UNIX tools
- While not always easy to setup, you did very well to use it efficiently

Elements to improve for next year

- Nothing much to improve in our opinion. Do you agree?

We would like to push these recommandations/tools in the very first weeks of your studies so you can use these tools from the beginning and in all courses. We will see if it is possible.

Part 1: Input/output processing

Git, GitHub and Markdown

Find this chapter on GitHub

Key points to remember for this chapter

0. Set up your Git environment with SSH and signed commits
1. Open an issue to discuss the feature (written in Markdown)
2. Clone or fork the project with SSH and checkout to a new branch
3. Make your changes, commit and push them as often as you want
4. Resolve conflicts if any
5. Create the pull request and add details if needed
6. Other members review and approve if everything is OK
7. The work is merged and you can delete the branch or the fork

Elements to improve for next year

- Add tips to validate the configuration of Git using the `~/.gitconfig` file
- Simplify the process of adding students to the GitHub organization:
  - Students open an issue
  - We add them to the GitHub organization and the right team
  - They create a pull request to add themselves to the list of students and we merge if everything is OK
- Move the forking process to the " *"Go further"* section

Java, IntelliJ IDEA and Maven

Find this chapter on GitHub

Key points to remember for this chapter

- Java and the JVM
- Maven as a build tool
  - `pom.xml` file
  - Plugins vs. dependencies
- Sharing code with Git and GitHub
  - `.gitignore` files
  - IntelliJ IDEA (and IDEs in general) configuration files
- Managing multiple versions of Java/dependencies with SDKMAN!

Elements to improve for next year

- Add a note regarding memory management with Java (garbage collector, etc.)
- Make students install older LTS versions instead of newer, non-LTS versions so they don't pollute their computer with unnecessary Java versions
- More details on how picocli works maybe?

Java IOs

Find this chapter on GitHub

Key points to remember for this chapter

- Java IOs classes and use cases (text files vs. binary files)
- Those 🤬 charset encodings (Unicode vs. UTF-8 vs. other charsets)
- Buffering and flushing
- End of line characters (`\n` , `\r` , `\r\n` , etc.)
- Dealing with exceptions (try-with-resources, etc.)

You **should always specify encodings and end of line characters explicitly** as the defaults are dependent on the platform your code is running on. Marking them explicitly will make your code **portable**.

Elements to improve for next year

- Nothing much to improve in our opinion. Do you agree?

Docker and Docker Compose

Find this chapter on GitHub

Key points to remember for this chapter

- Bare-metal vs. virtualization vs. containerization
- Docker as a containerization tool
  - Images
  - Containers
  - Registries
  - Dockerfile files
- Docker Compose as a tool to manage multiple containers
  - Docker Compose files

Elements to improve for next year

- Add elements to publish ports with Docker (using the `-p` / `--publish` parameters)

It is still a quite difficult and abstract chapter to teach. We did try our best to give you all the elements needed to understand this topic incrementally so you would not be overwhelmed.

Practical work 1

Find this practical work on GitHub

What you were ask to do

- A CLI to process files
- Use Java, Maven and picocli
- You can choose what the CLI will do
- Practice a professional Git workflow and publish your CLI on GitHub for others to discover and use

We have seen some very interesting projects! Caesar cipher, image processing, JSON-XML convertor, etc.

Do not hesitate to share your project and to continue to work on it!

Elements to improve for next year

- More time for the presentations

Part 2: Network programming with TCP and UDP

Define an application protocol

Find this chapter on GitHub

Key points to remember for this chapter

Defining an application protocol is **not an easy task**. There are many ways to do it, and there is no "one size fits all" solution. You should always keep in mind the following points:

1. Context - What the application protocol is used for?
2. Transport - What protocol(s) is/are involved? On which port(s)? How are messages/actions encoded? How are messages/actions delimited? How are messages/actions treated? Who initiates/closes the communication? What happens on an unknown message/action/exception?

3. Messages/actions - What are the messages/actions? What are the parameters? What are the return values? What are the exceptions?

   Always try to describe these **for a given context**, not from each point of view (e.g. "*making an order*" with the input/outputs from the client to the server and the responses instead of "*the client sends these messages and the server replies these messages with these outputs*"). It makes it *way* easier to understand and to implement.

4. Example diagrams - What are the examples of messages/actions? What are the examples of exceptions? Illustrate your application protocol with diagrams to make it easier to understand.

Elements to improve for next year

- Nothing much to improve in our opinion. Do you agree?

Java TCP programming

Find this chapter on GitHub

Key points to remember for this chapter

- TCP as a reliable protocol
- TCP is unicast only
- A client and a server communicate using `Socket` and `ServerSocket` classes on a given port for a given host
- Dealing with sockets' streams are the same as with files: buffering, flushing, charsets, end of line characters, exceptions handling, etc.

Elements to improve for next year

- Nothing much to improve in our opinion. Do you agree?

Java UDP programming

Find this chapter on GitHub

Key points to remember for this chapter

- Differences between TCP and UDP
- Reliability of UDP
- Unicast, broadcast and multicast
- A client/server (unicast) and or emitter/receiver (multicast) communicate with the `DatagramSocket`, `DatagramPacket` and `MulticastSocket` classes
- Messaging patterns
- Service discovery protocols

Elements to improve for next year

- Nothing much to improve in our opinion. Do you agree?

Java network concurrency

Find this chapter on GitHub

Key points to remember for this chapter

- Concurrency allows to manage multiple task simultaneously
- Java offers classes and data structures to manage concurrency
- Multiple strategies exist to manage concurrency, with different trade-offs:
  - Unlimited threads can use all the available resources
  - Threadpools can set the maximum number of threads but need to calculate the right amount to use resources efficiently
  - Recommended: Threadpools for Java < 19 and VirtualThreads for Java > 19

Elements to improve for next year

- Nothing much to improve in our opinion. Do you agree?

SMTP and ncat

Find this chapter on GitHub

Key points to remember for this chapter

- Differences between SMTP, POP3 and IMAP
- SMTP security concerns
- How is an email sent and received from one client to another through multiple SMTP servers
- Using a SMTP mock server to test your application

Elements to improve for next year

- Move this chapter during the practical work 1 presentations

Practical work 2

Find this practical work on GitHub

What you were ask to do

- A network application using TCP and/or UDP with its own application protocol
- You can choose what the network application will do
- Your first experiments with Docker to publish your application to the GitHub Container Registry

We have seen some very interesting projects! Chat applications, hanging man (poor boy...) games, naval battle games, etc.

Do not hesitate to share your project and to continue to work on it!

Elements to improve for next year

- The usage of Docker for some applications was a bit hard. We might need to improve the tips regarding this point.

Part 3: Network programming with HTTP

SSH and SCP

Find this chapter on GitHub

This chapter will not be in the exam! The other classe(s) did not study this topic.

Key points to remember for this chapter

- Acquire a real remote server on a cloud provider (Azure)
- SSH keys and how to clone/sign commits using Git
- SSH keys and how to connect to a remote server
- SCP and how to copy files to a remote server

Elements to improve for next year

- Nothing much to improve in our opinion. Do you agree?

HTTP and curl

Find this chapter on GitHub

Key points to remember for this chapter

- HTTP request methods and their response status codes
- HTTP path parameters, query parameters and body
- HTTP headers
- HTTP content negotiation
- Structure of a HTTP request/response (raw HTTP)
- HTTP sessions using a query parameter or a cookie
- API design

Elements to improve for next year

- Add a bit of content regarding HTML, JavaScript and CSS
- Refer to official documentation to generate full web applications (UI + API) using the official resources to help for the BDR + DAI project:
  - https://javalin.io/plugins/rendering
  - https://javalin.io/tutorials/
- I would love to go deeper on this topic but the time is limited. You will learn more about this topic in future courses

Web infrastructures

Find this chapter on GitHub

Key points to remember for this chapter

- Functional and non-functional requirements
- How HTTP features can be used to build web infrastructures:
  - The `Host` header
  - Reverse proxy
  - System scaling
  - Load balancing

Elements to improve for next year

- Improve the diagrams

Caching and performance

Find this chapter on GitHub

Key points to remember for this chapter

- How HTTP features can be used to implement caching and improve performance of web applications:
  - Expiration model
  - Validation model
- Different types of cache
- Where to cache

Elements to improve for next year

- Improve diagrams with a database actor
- Improve diagrams when the client asks for a content that has been changed for the validation model examples

Practical work 3

Find this practical work on GitHub

What you were ask to do

- Acquire and configure a virtual machine on the cloud
- Install Docker and Docker Compose on the virtual machine
- Develop a simple CRUD API to manage resources
- Deploy the applications (reverse proxy + CRUD API)
- Access the applications from a (free) domain name

We have seen some very interesting projects! Pokédexes, GPG keys management, etc.

Do not hesitate to share your project and to continue to work on it!

Elements to improve for next year

- Better define the expectations between the DAI course and BDR course.
- Nothing else to improve in our opinion. I think it is the perfect practical work to close the loop of the course. Do you agree?

Course materials

Elements to improve for next year

- Remove the PDFs generations to ensure all students use the latest available course material on GitHub
- Merge the code examples and solutions repositories in the main repository to help management and access for everyone
- Read proof all examples and normalize without the usage of interfaces but their concrete class implementation to avoid confusion
- Try out a new structure for the calendar/planning: PR #560
- Make usage of a quiz application to kickoff each course session

Practical works

Elements to improve for next year

- Improve the deadlines and expectations related to the submissions: PR #570
- State the presentation time/question time more precisely
- Add milestones from weeks to weeks to follow the advancement

Evaluations

Elements to improve for next year

- We will allow two double-sided sheets (so four single-sided sheets) of personal notes for each evaluation to avoid confusion on the number
- Some evaluations were a bit too long. We will try to improve that in the future
- Some questions are still too vague
- Some evaluations are still too long

GAPS evaluation

Let's check these *beautiful* charts!

You can find all Framasoft and GAPS evaluations on the repository.

Remaining questions

Here are some remaining questions we never take the time to discuss and I would like your opinion on them. There will be no judgement and no consequences. I just want to understand your point of view.

- Did you find the course too difficult or having too much content?
- Did you feel the need to cheat during the semester?
- Is there anything that you feel was not fair during the semester?
- Do you have a remaining question you would like to ask?

Exam preparation

What you need to know for the exam

Exam preparation

Where and when?

- **Date**: 31.01.2025 (Friday)
- **Time**: 08:30
- **Duration**: 60 minutes
- **Place(s)**: G01, H01, H02 and J01

Find all examens on SACHEM.

What to expect?

A paper exam that will test your knowledge on everything we have seen during the semester (theoretical content, practical content and practical works).

You will be asked to read/write/understand some code.

You must be ready as there will be no time to lose.

Allowed resources?

Two double-sided sheets (so four single-sided sheets) of personal notes.

What to do during the exam?

Read the exam carefully.

Always try to write something.

You all have the skills to pass the exam.

How to revise?

All typical questions are at the end of each chapter you studied.

All previous evaluations are available in the course repository:

- Evaluation 1
- Evaluation 2
- Evaluation 3

Questions

Do you have any questions?

Conclusion

“ *This course defines the basics of network communication and how all these communications are programmed.*

At the end of the course, you will know how to define, program and deploy network applications, how to interact with them, and the different elements to pay attention to make robust applications.

Whether you are in software, security, data science, embedded or network, you will have to deal with network applications (APIs, devices, etc.). This course will give you a solid grounding in this world. ”

You did it! Congrats!

You can be proud of yourself! You all did a great job! We had a blast following your progress during this semester.

This course is now over, but we hope you enjoyed it and learned a lot.

This course is part of a larger curriculum, and we hope you will be able to apply what you learned in the next courses.

This course is only the start of what you can learn about. We hope you will continue to learn about these topics in the future.

How to stay up to date?

Staying up to date is a challenge in the IT world. Here are some resources to help you with content related to this course:

- https://news.ycombinator.com/
- https://www.reddit.com/r/programming/
- https://www.reddit.com/r/linux/
- https://www.reddit.com/r/selfhosted/
- https://github.com/awesome-selfhosted/awesome-selfhosted

Well, what now?

You have gained significant knowledge during this course. Use this knowledge wisely. With great power comes great responsibility. You can now:

- Go deeper in the web development world
- Enter the game of self-hosting
- Interact with the open-source community

Thrive to learn more, and do not hesitate to share your knowledge with others.

Closing remarks

My personal closing remarks are the following:

- **Always ask yourself the right questions**: *Why? How? What? Am I doing the right thing for the right cause? Do I listen to myself?*
- **Always do what is good for you**: Health, relationships, friends are more important than work! *You* are more important than work!
- **Always trust yourself and your guts**: Do what you think is right!
- **Help others, be kind**: Cooperation is better than competition!
- **Stay critical**: Your opinion matters and can make a difference.

Acknowledgements

I would like to thank the following people for their help on this course (in no particular order):

Hadrien Louis (2023-2025), **Géraud Silvestri** (2025-2026) and **Camille Koestli** (2025-2026) who helped me to prepare this course.

Juergen Ehrensberger (2023-2025) for sharing the teaching with the other classes.

And of course, **you**, for your participation and your interest in this course! It was a pleasure to teach you and I hope to see you again!

Thank you, good luck, and farewell!

Apéro time! 🎉

Sources

- Main illustration by MChe Lee on Unsplash
- Illustration by Kenny Eliason on Unsplash
- All other illustrations' sources are available in their respective course materials