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* ❤️

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 (written with Markdown) to discuss the feature
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 or update 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

- For people on Windows, create a document to help set up a proper and working environment (WSL2, Git, Java, Maven, SDKMAN!, etc.)
- Add more information about SSH and signed commits maybe?

Java, IntelliJ IDEA and Maven

Find this chapter on GitHub

This chapter will not be in the exam! The other classes did not study this topic as deeply as you did.

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

- Make usage of picocli with a concret example (introduction to picocli and its documentation) instead of Logback
- Make usage of Spotless to format the code for uniformity

Java IOs

Find this chapter on GitHub

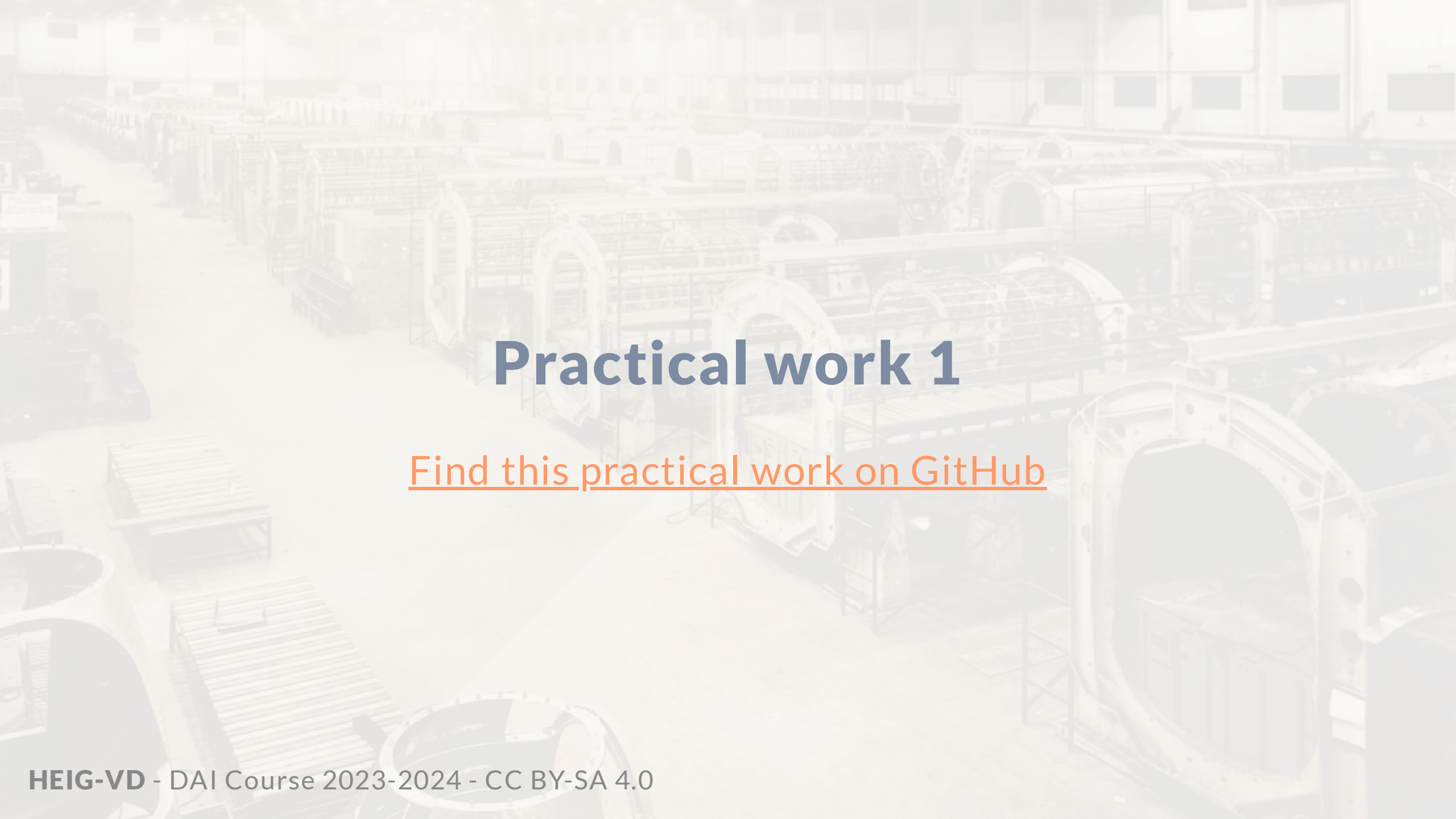
Key points to remember for this chapter

- Java IOs classes and use cases (text files vs. binary files)
- Those  charset encodings (Unicode vs. UTF-8, 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

- Add disclaimer about the `PrintWriter` class - it is not recommended to use it as it swallows exceptions and does not throw them



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
- Publish your CLI on GitHub

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

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

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 (UML, sequence, etc.) to make it easier to understand.

Elements to improve for next year

- Remove the exploration in details of known protocols (SMTP, POP3, IMAP, SSH, etc.) and focus on the definition of an application protocol
- Add more examples of application protocols (e.g. a protocol to order a pizza, a protocol to play a game, etc.)
- Add more examples of diagrams

This was a **very** difficult chapter to teach. I hope you understood the importance of defining an application protocol and how to do it. If you can, use an existing protocol instead of defining your own.

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

- Move this chapter after the Java TCP programming chapter
- More details on each line of the Dockerfile and Docker Compose file maybe?
- Use more concrete examples instead of a generic Alpine image with greetings, `tree` command with volumes and File Browser as a web application with ports maybe? It is hard to think of a good example that is not too complex to understand and not too simple to be useless.

SMTP and Telnet

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 after the Java TCP programming chapter
- Make usage of ncat (part of nmap) instead of Telnet - Rename the chapter to "SMTP and ncat"

SSH and SCP

Find this chapter on GitHub

This chapter will not be in the exam! The other classes did not study this topic.

Key points to remember for this chapter

- 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

- Maybe this chapter is too abstract at the time of the semester? Maybe it should be moved to the end of the semester? With the web infrastructures chapter?

Java TCP programming

Find this chapter on GitHub

Key points to remember for this chapter

- TCP as a reliable protocol
- Write a client and a server that communicate using `Socket` and `ServerSocket` classes
- Deal with sockets' streams (deal as with files - buffering, flushing, charsets, end of line characters, exceptions handling, etc.)
- Use an executor (`ThreadPool` is recommended but you should know other executors and their pros and cons) to handle multiple clients

Elements to improve for next year

- Move this chapter right after the Define an application protocol chapter so that the practical work can be done in parallel with the next chapters
- Add details/examples on variable-length messages/actions (e.g. `readLine()` vs. `read()` with a fixed buffer size)
- Add a proper introduction to concurrency and its issues
- Make an example built from scratch to show how to use the `Socket` and `ServerSocket` classes with multiple clients (with `ThreadPool` executor) and then show the examples for other implementations

Practical work 2

Find this practical work on GitHub

What you were ask to do

- A TCP network application with its own application protocol
- You can choose what the network application will do
- Groups of two students

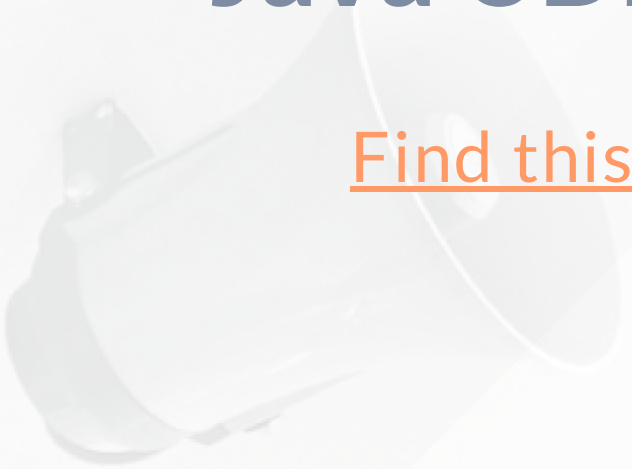
We have seen some very interesting projects! Chat applications, hanging man (poor boy...) games, naval battle games, etc.

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

Elements to improve for next year

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

Java UDP programming



Find this chapter on GitHub

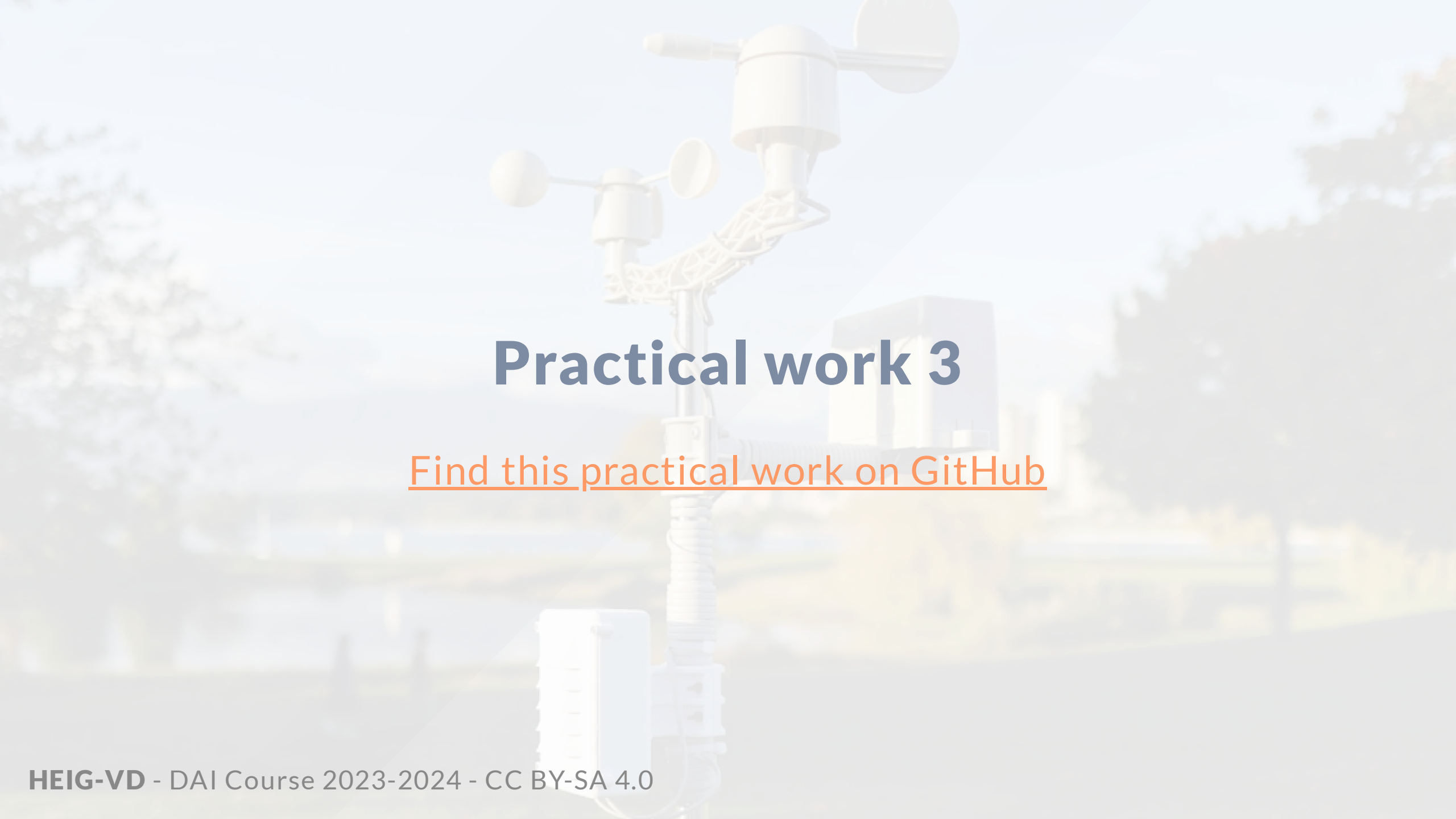
Key points to remember for this chapter

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

Elements to improve for next year

- Make an example built from scratch to show how to use the `DatagramSocket`, `DatagramPacket` and `MulticastSocket` classes with multiple clients (with `ThreadPool` executor) and then show the examples for other implementations
- Add example for the request/response pattern with unicast for the practical work

This was a **very** difficult chapter to teach. I hope you understood the main differences between TCP and UDP and how to use UDP.



Practical work 3

Find this practical work on GitHub

What you were ask to do

- An UDP network application
- You can choose what the network application will do
- Share your application on GitHub with Docker and Docker Compose
- Groups between 2 and 3 students

We have seen some very interesting projects! Teletext application, multiplayer Pong game, Tower defense game, IoT logging system, etc.

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

Elements to improve for next year

- Improve the criteria for the evaluation of the practical work (some are too vague)

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 (stateless vs. stateful)
  - Using a query parameter or a cookie
- API design

Elements to improve for next year

- Nothing much to improve in my opinion. I tried to improve the balance between the theoretical/practical content and I think it is quite good now. Do you agree?

Web infrastructures

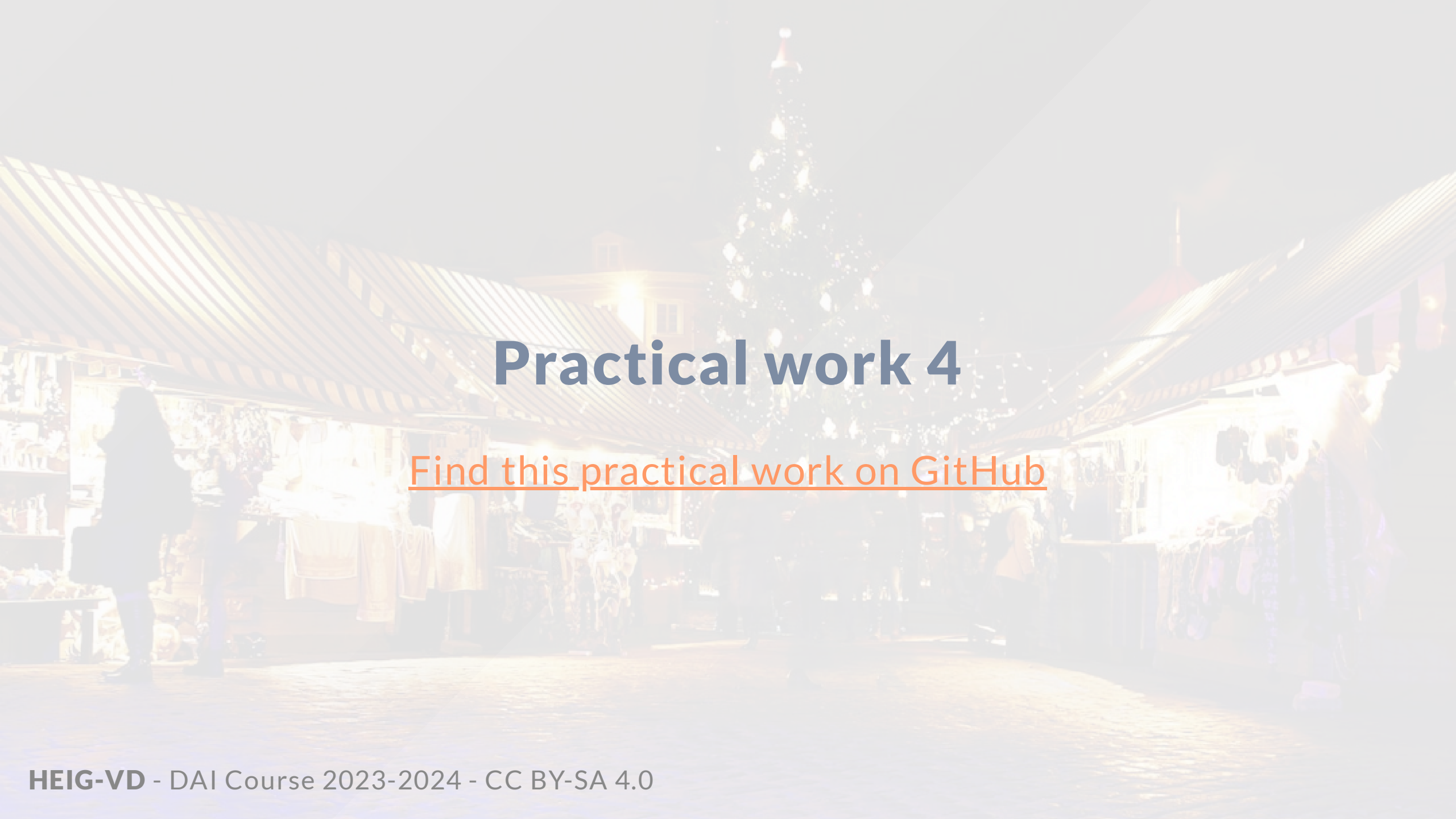
Find this chapter on GitHub

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
  - Caching

Elements to improve for next year

- Nothing much to improve in my opinion. I tried to improve the balance between the theoretical/practical content and I think it is quite good now. Do you agree?



Practical work 4

Find this practical work on GitHub

What you were ask to do

- Get and access a virtual machine on our cloud with SSH
- 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
- Groups of 4 students

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

Elements to improve for next year

- Make a common project with the BDR course to have a database to manage (and make a more realist application) maybe?
- Nothing else to improve in my opinion. I think it is the perfect practical work to close the loop of the course. Do you agree?

Evaluations

Elements to improve for next year

- Most of them were too long (especially the first two)
- Some questions were too specific
- It was the most painful point, we will try to improve this!

Course materials

Elements to improve for next year

- The PDF generation is okayish but not perfect
- Make a better mix between theoretical and practical content (more like the last two chapters)

GAPS evaluation

Let's check these *beautiful* charts!

You can find all GAPS evaluations on the repository.

"J'ai moins apprécié et je propose..." (Résumé)

Évaluations

- Difficulté à réviser pour des tests théoriques.
- Améliorer la durée des tests.
- Clarifier les évaluations.
- Réduire le nombre d'évaluations ou de laboratoires.
- Suggestion d'autres formes d'évaluation (mini-évaluations similaires à des entretiens d'embauche, deux évaluations mais avec des évaluations informatives entre les deux, etc.).

Travaux pratiques

- La liberté créative peut être décourageante, surtout si elle ne correspond pas aux critères.
- Ne voit pas l'intérêt de la partie orale après les labos, trouve les présentations stressantes.
- La quantité de travail dans les labos est jugée trop importante.
- Suggestion d'une consigne de base pour tous, avec la possibilité de discuter d'autres sujets.

Méthodologie et support de cours

- Besoin d'explications détaillées sur les codes pratiques, sans y consacrer trop de temps.
- Préférence pour des cours plus concis et guidés, moins de texte pour une assimilation plus efficace.
- Le temps de travail semble excessif par rapport à la valeur perçue du cours.
- Manque de préparation pratique malgré la présentation des concepts.

"J'ai particulièrement apprécié..." (Résumé)

Évaluations

- Appréciation de la plateforme d'évaluation en ligne pour reproduire des conditions réelles.
- Soutien à l'idée d'avoir plus de tests, mais sur des sujets moins étendus.

Travaux pratiques

- La possibilité de choisir ses projets est motivante.

- Apprécie la présentation des labos, mettant en valeur le travail effectué et développant les compétences en présentation.
- Les labos sont bien appréciés.
- Les évaluations pratiques, ainsi que leurs présentations, sont bien réalisées.

Méthodologie et support de cours

- Soutien de l'idée d'avoir moins de théorie et plus de pratique pour un apprentissage efficace.
- Apprécie la présence des enseignants et de l'assistant, même en cas de problèmes.

- Excellentes explications fournies lors des questions.
- Cours dispensé sur une plateforme professionnelle.
- Appréciation des supports de cours, régulièrement mis à jour.
- Appréciation d'avoir la structure et la planification du cours dès le début.
- Appréciation de l'ambiance en classe, du confort ressenti, et de l'attention portée à chacun-e.
- Appréciation de l'écoute et de la prise en compte immédiate des remarques des étudiant-e-s.
- Appréciation de la possibilité de corriger des aspects dérangeants, soulignant que le cours n'est pas figé et les enseignants accessibles.

Taboo questions

Here are some taboo 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?
- 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 taboo question you would like to ask?

Exam preparation

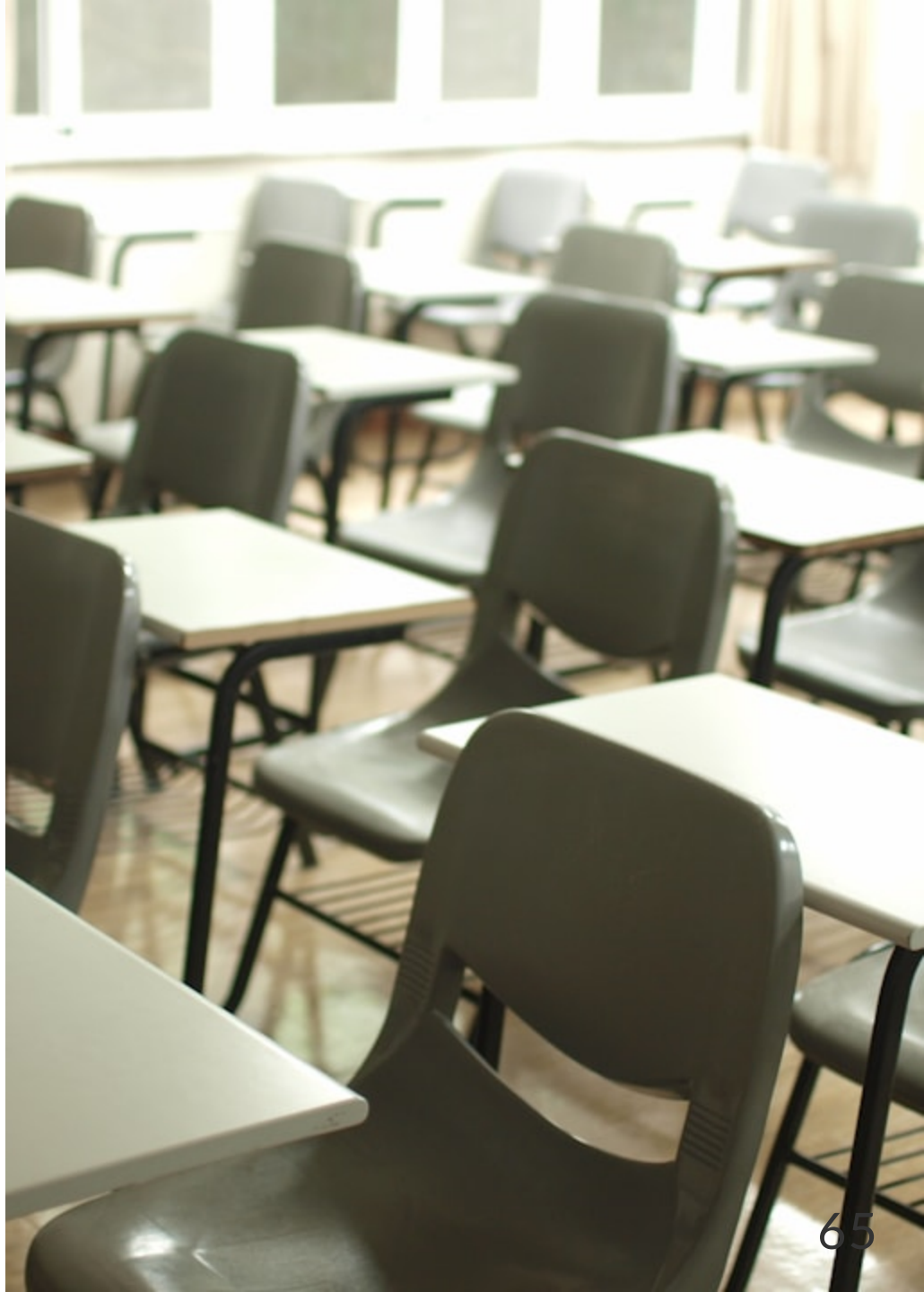
What you need to know for the exam

Exam preparation

Where and when?

- **Date**: 07.02.2024 (Wednesday)
- **Time**: 08:00
- **Duration**: 1 hour
- **Place(s)**: G01, J01 and H01

Find all examens on SACHEM.

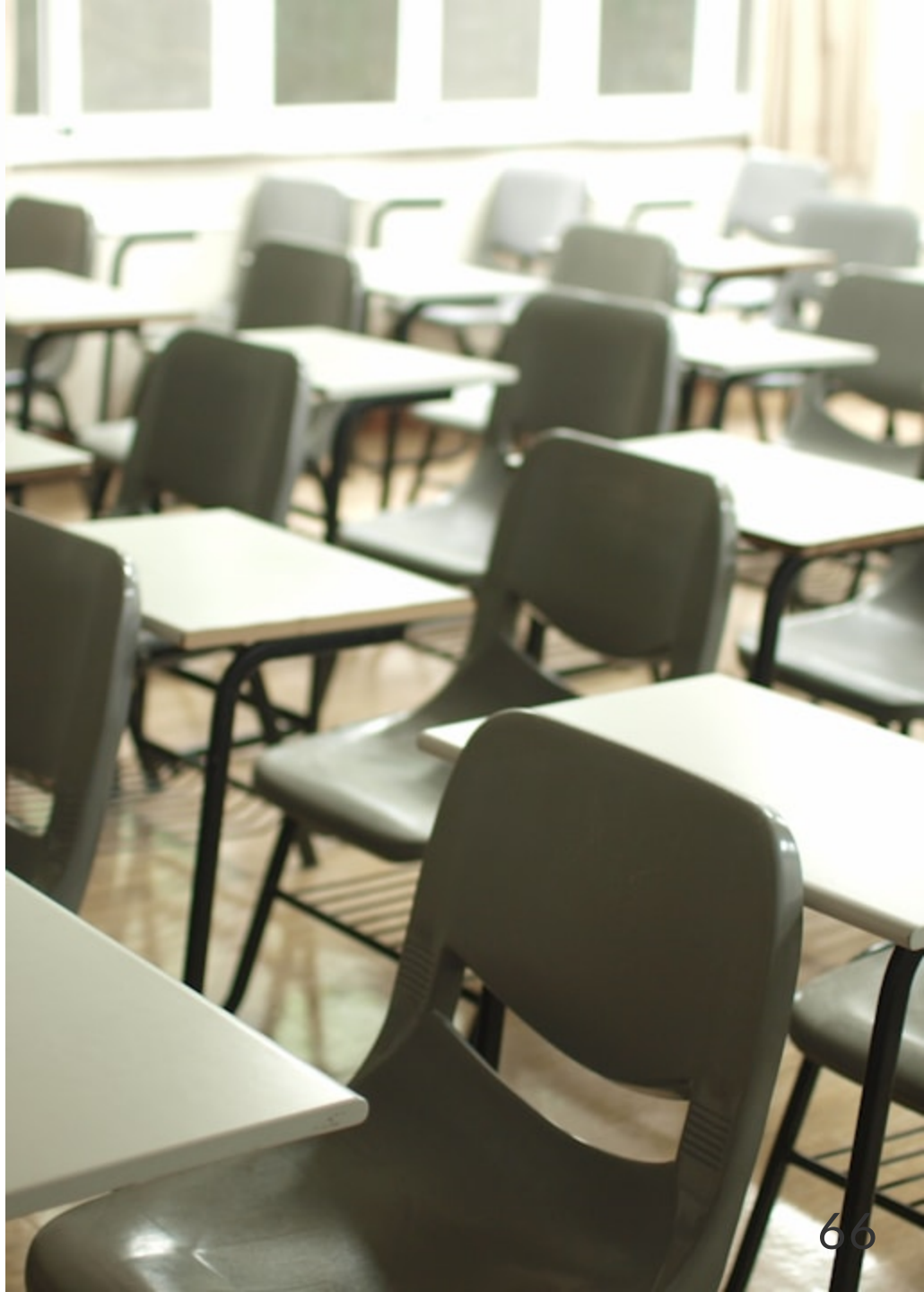


What to expect?

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?

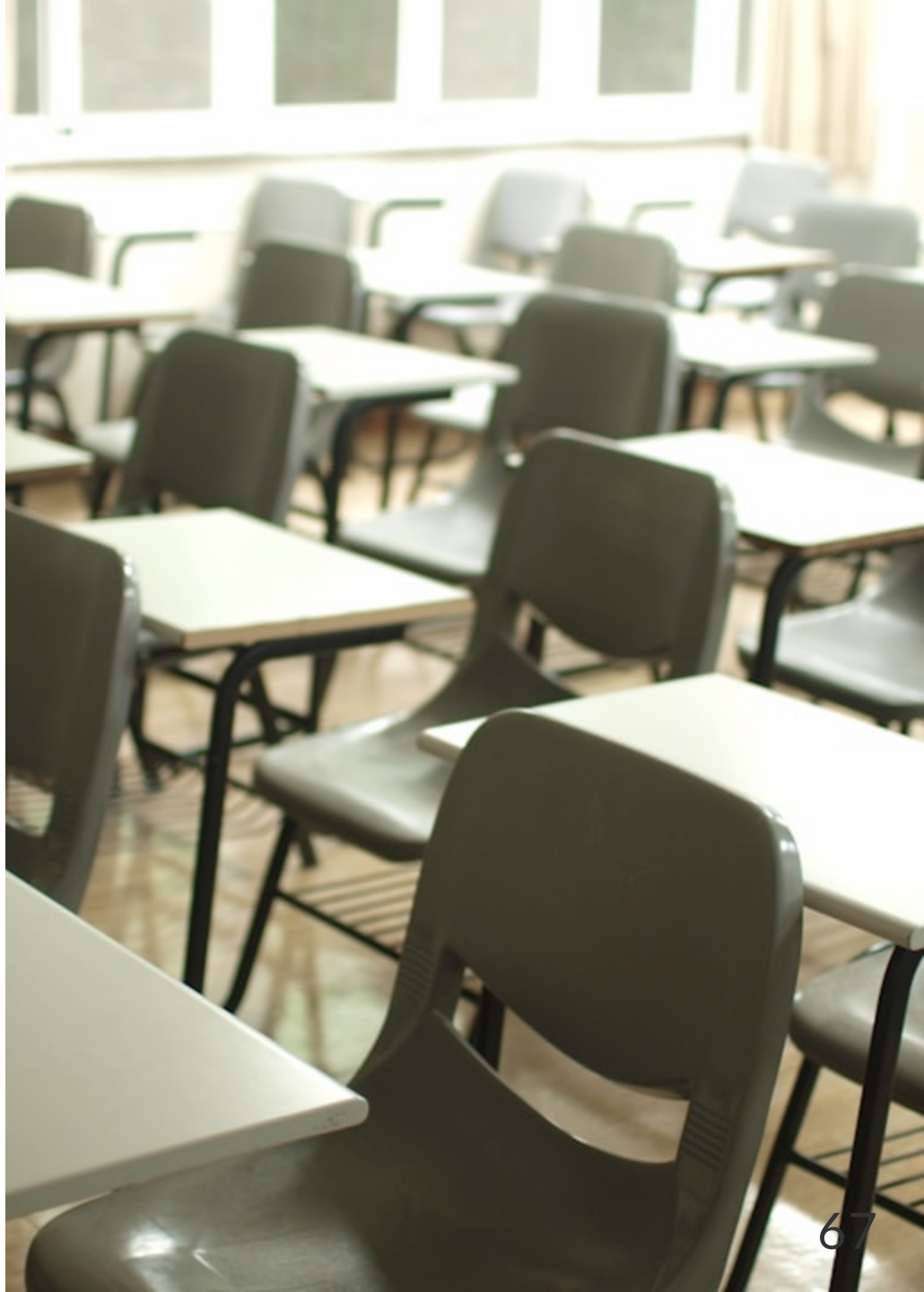
Two double-sided sheets (so four single-sided sheets) of personal notes.

What to do during the exam?

Read the exam carefully.

Always try to write something.

You all have the skills to pass the exam.

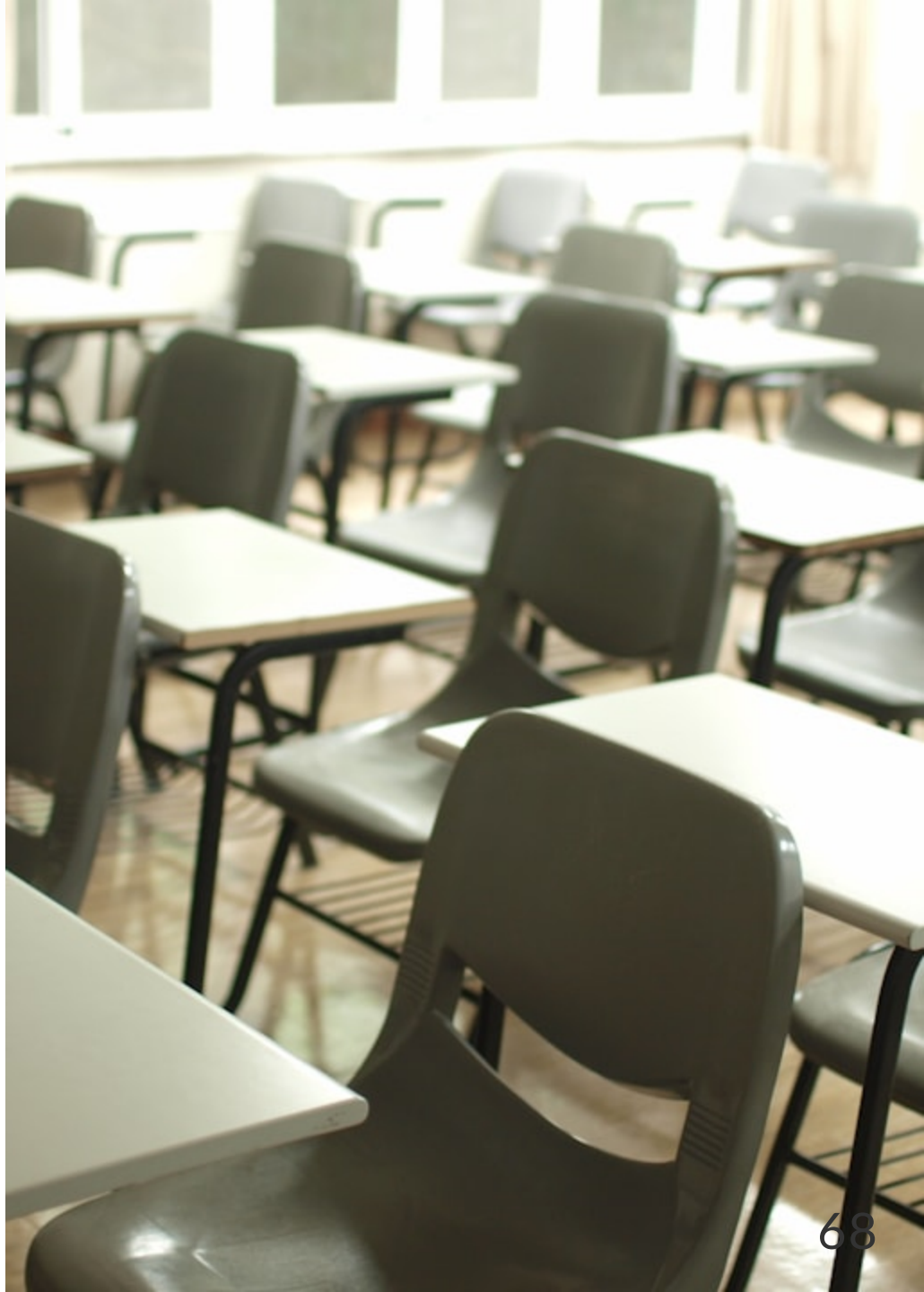


How to revise?

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
- Evaluation 4



Questions

Do you have any questions?

Conclusion

Conclusion

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/homelab/
- https://www.reddit.com/r/programming/
- 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 stay critical**! Don't believe everything you read or hear!
- **Always ask yourself 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!

Thank you, good luck, and farewell!

Apéro time! 🎉

Sources

- Main illustration by MChe Lee on Unsplash
- Illustration by Kenny Eliason on Unsplash
- Illustration by Roman Synkevych on Unsplash
- Illustration by Nathan Dumlao on Unsplash
- Illustration by Martijn Baudoin on Unsplash
- Illustration by Birmingham Museums Trust on Unsplash
- Illustration by Iñaki del Olmo on Unsplash
- Illustration by CHUTTERSNAP on Unsplash

- Illustration by Joanna Kosinska on Unsplash
- Illustration by Mathew Schwartz on Unsplash
- Illustration by Carl Nenzen Loven on Unsplash
- Illustration by Rafael Rex Felisilda on Unsplash
- Illustration by Possessed Photography on Unsplash
- Illustration by Jorge Ramirez on Unsplash
- Illustration by Ashley Knedler on Unsplash
- Illustration by Nicolas Picard on Unsplash
- Illustration by Lāsma Artmane on Unsplash