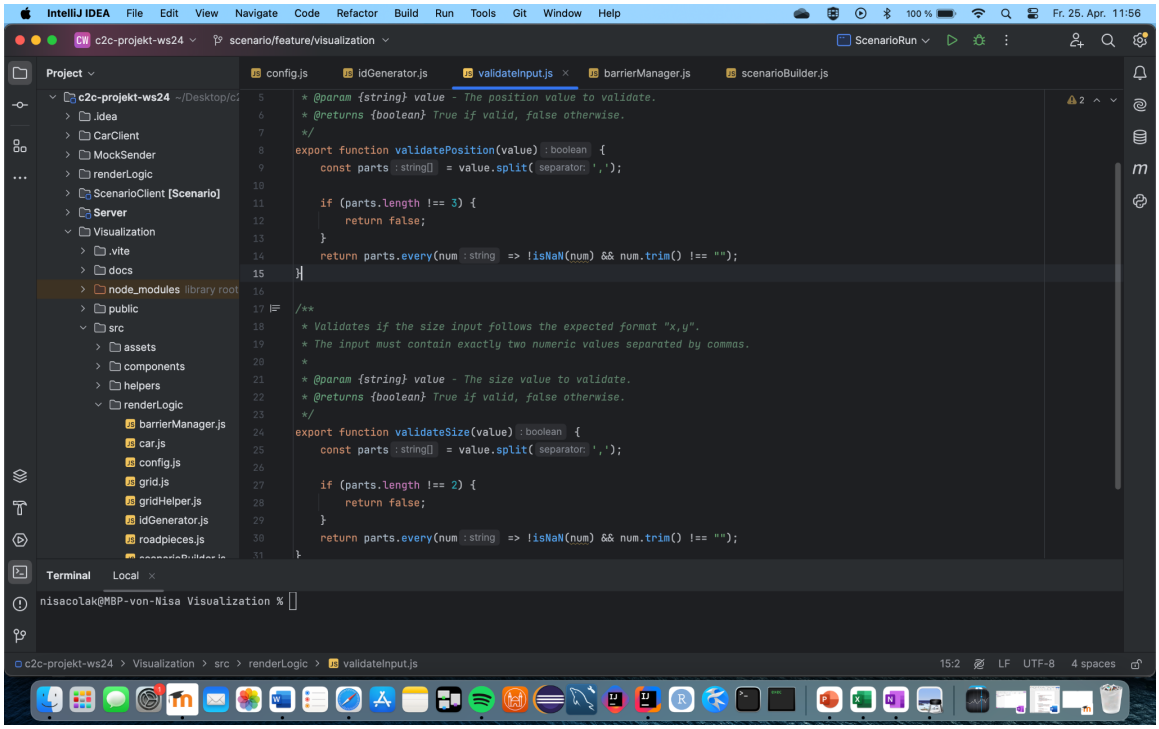
## Exercise 1 – Task 1: Installing an IDE

Chosen IDE: IntelliJ IDEA

To complete the programming tasks, the integrated development environment IntelliJ IDEA was installed and successfully set up. IntelliJ IDEA offers many helpful features for software development, especially for languages such as Java, JavaScript, Kotlin, and more.

Screenshot of the opened IDE:




1. 10 Advantages of an IDE (e.g., IntelliJ IDEA) Compared to a Simple Text Editor:

2.Syntax highlighting – makes code more readable and helps identify errors.
3.Automatic code completion – saves time and reduces typing mistakes.
4.Real-time error detection – shows potential issues while typing.
5.Integrated debugger – allows step-by-step execution with breakpoints.
6.Version control with Git – built-in Git support for commit, push, pull, etc.
7.Code navigation – easy access to definitions, references, and implementations.
8.Refactoring tools – automatic renaming, method extraction, and more without errors.
9.Project management – clear handling of project structure, modules, and dependencies.
10.Plugin system – extensible with many plugins (e.g., for Docker, databases, testing).
11.Unit test integration – support for testing frameworks like JUnit or Jest.

## Exercise 1 -Task 2: Ulam Function

Although a formal mathematical proof for all integers is still missing, extensive numerical calculations have shown that the Ulam function reaches 1 for all tested numbers, even for very large values (up to 10201020 and beyond). This includes all integers less than 1 million.

The Java code above verifies that all numbers less than 1 million eventually reach 1. Therefore, based on this numerical evidence, we can conclude that the Ulam function will terminate at 1 for all integers nn such that $1 \leq n < 1061 \leq n < 106$.

## Exercise 1 -Task 3: GitHub

- Account erstetellt und auch eine eigene repository

## Exercise 1 – Task 4 : Install MagicDraw UML


## Exercise 1 – Task 5:  Class Diagram Developer

- Klassen Diagram hab ich gemacht

- Code hab ich auch zum Klassen diagram
  - bei dem Objektdiagramm hatte ich probleme

## Exercise 1 – Task 6: Define the term software design. Explain how this differs from software analysis. Define the term software design. Explain how this differs from software analysis.

- Software design is the process of planning and defining the structure, components, interfaces, and data for a software system to satisfy specified requirements.

- Software analysis, on the other hand, focuses on understanding and documenting what the system should do, by gathering and analyzing requirements.
  In short, software analysis defines the "what," while software design defines the "how.

## Exercise 1 – Task 7: Explain why a software design is necessary for a software project. Can you think

A software design is necessary because it provides a clear blueprint for developers to follow, ensuring that the system is built correctly, efficiently, and is maintainable.
Without a proper design, a project may become chaotic, suffer from integration problems, be difficult to scale, or even fail altogether.
For example, if you try to build a complex e-commerce platform without first designing the system, you could end up with incompatible modules, security flaws, and a confusing user experience.

- Personio is a cloud-based SaaS platform designed to streamline and automate HR processes for small and medium-sized enterprises (SMEs). It offers a modular suite of tools covering the entire employee lifecycle, from recruitment to offboarding, with a strong emphasis on automation, integration, and compliance.

- **Exercise 1 – Task 8: Are the design activities of architectural design, database design, user interface design and component design independent or interdependent? Using an example, explain why.**

The design activities — architectural design, database design, user interface design, and component design — are interdependent.
Changes in one area can significantly affect the others.
For example, if the database design changes (e.g., new data relationships are introduced), the user interface and the component logic must also adapt to these changes.
An online banking system illustrates this: if you decide to change how user accounts are stored in the database, you will also need to modify the way the user interface displays account information and how back-end components process transactions.