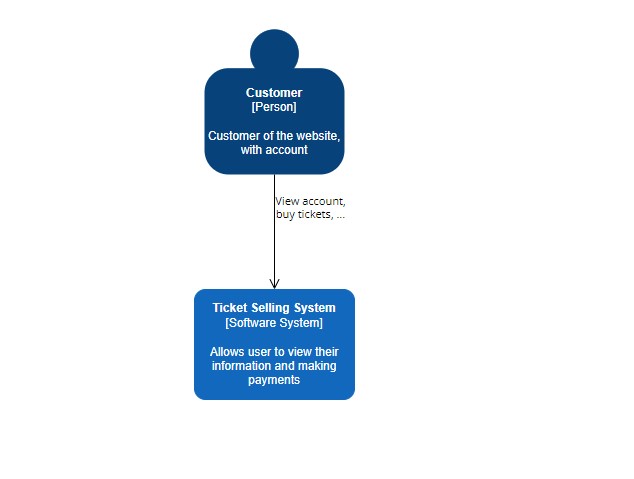
Hristo Kolev

C4 model arcitecture

**C1: System Context diagram**

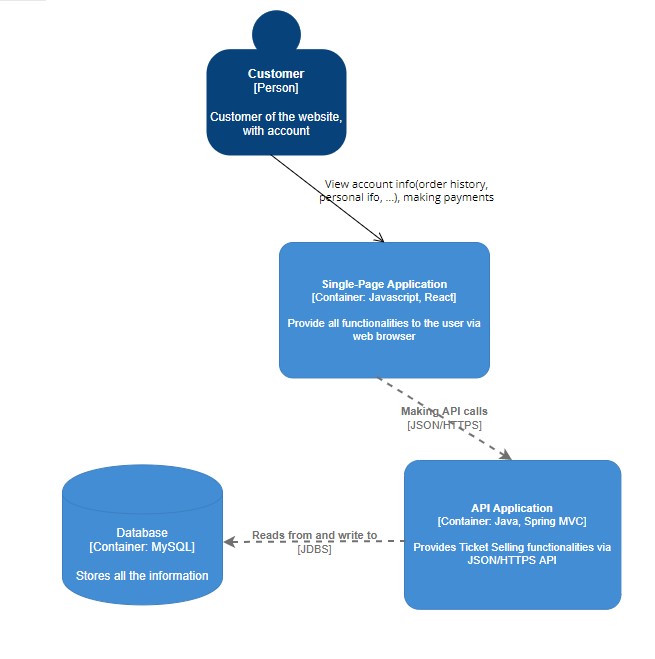


**Primary element**: The software system in scope

This diagram shows the system surrounded by its users that interact with. Focus should be on People (roles, actors, personas, etc.) and Software system rather than technologies, protocols, and other details.

**C2: Container diagram**

**Primary element**: Container within the software system in the scope

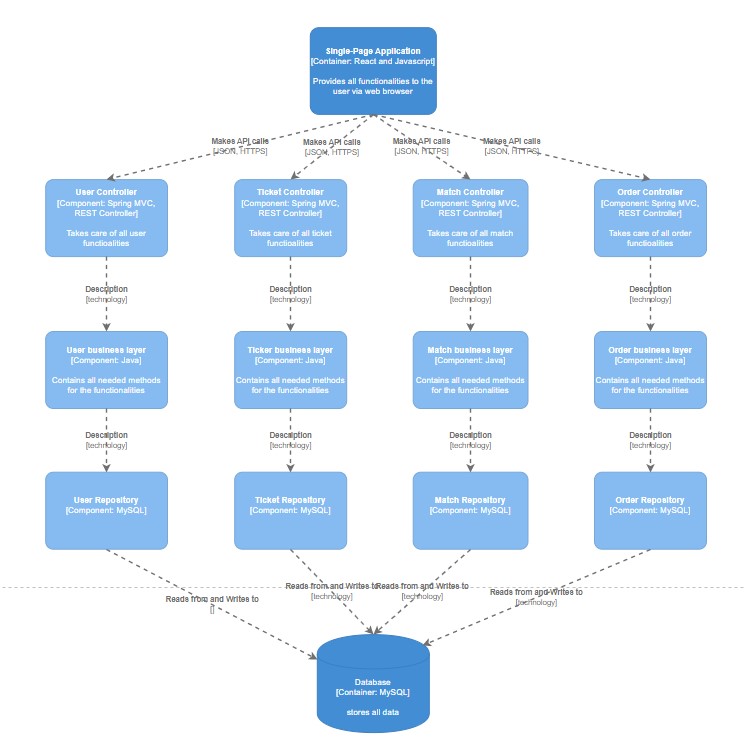


A “container” is a Single-page application, data schema, file system. The container is a separately runnable unit that execute code or store data.

The Container diagram shows the high-level shape of the software architecture and how responsibilities are distributed across it. It also shows the major technology choices and how the containers communicate with one another. It's a simple, high-level technology focused diagram that is useful for software developers and support/operations staff alike.

**C3: Component Diagram**

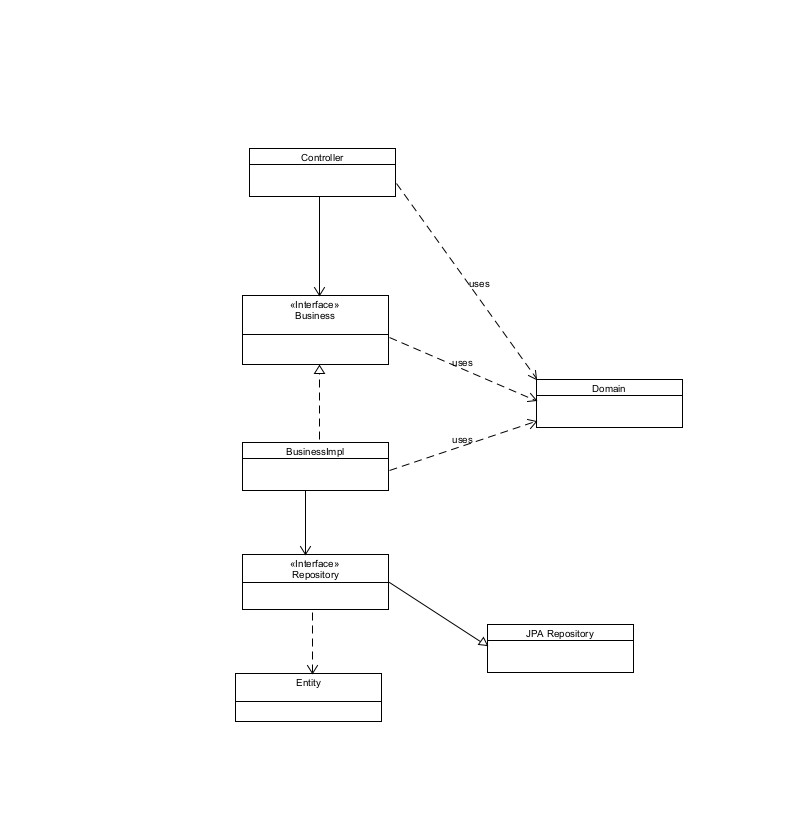
**Primary element**: Component within the container in the scope



On this diagram you can see zoomed in each container further to identify the major structural building blocks. The diagram is made up of a number of “components”, what these components are, their responsibilities and the technology/implementation details.

**C4 Code Diagram**

**Primary elements**: Code elements (e.g. classes, interfaces, objects, functions, database tables, etc.) within the component in scope.

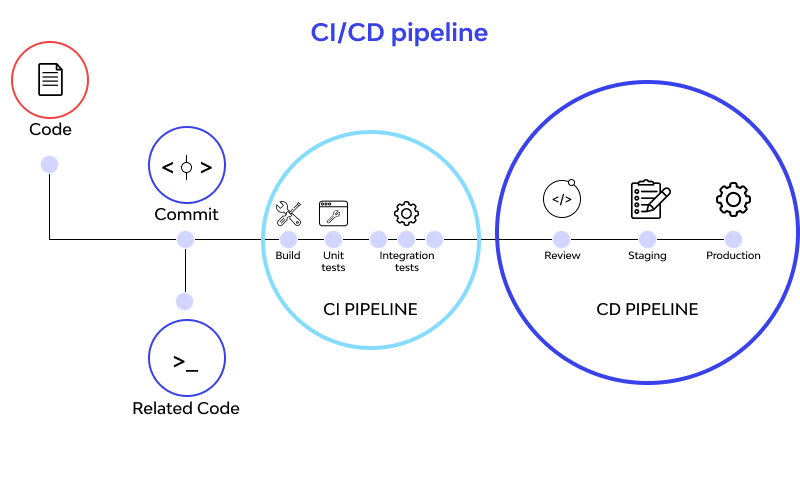


This is a zoomed in diagram of each component to show how is it implemented as code (using UML diagrams, entity relationship diagrams or similar).

**Intended audience**: Software architects and developers.

**Recommended for most teams**: No, for long-lived documentation, most IDEs can generate this level of detail on demand.

Design Document



CI/CD is a culture, operating principles and set of practices that application development teams use to deliver code changes more frequently and reliably. By automating integration and delivery, CI/CD lets software development teams focus on meeting business requirements while ensuring code quality and software security (Sacolick, 2022).

CI/CD stands for Continuous Integration and Continuous Delivery, which are two very different but related concepts that sometimes get used interchangeably. CI means new code changes to an app are regularly built, tested, and merged to a shared repository. It's a solution to the problem of having too many branches of an app in development at once. (*What Is CI/CD?*, n.d.)

**Why I use CI/CD?**

In agile projects the code change rapidly and if you sync up very often, it will become messy code and could be created different branches that can not be merged with the main one. Using this pipeline, the chance of error is minimum. This pipeline always checks for errors by building and testing the app and if

there is one it will inform you about it. For example, I use CI/CD pipeline in my git. When I upload my work the version control “Built” and “Test” the project. Build stage check if there is missing important dependency etc. Test stage is checking the unit tests. If there are errors I will get an email with information about the errors.

References:

Sacolick, I. (2022, April 15). *What is CI/CD? Continuous integration and continuous delivery explained*. InfoWorld. https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html

*What is CI/CD?* (n.d.). https://www.redhat.com/en/topics/devops/what-is-ci-cd