README.MD

Quality attributes

Although it is not intended that you go into design or implementation details here, the quality attributes described in this section are too general. Are there specific parts of the system for which performance is a concern? Why? What about availability and scalability, are they more important for some parts of the system than others? Why is that?

As these quality attributes are typically things that you want to measure and regularly monitor, it is likely that we would find them again in the "Operation" section of this report, describing how they are in fact measured/monitored.

The "Challenges and possible solutions" section right now is a lot about quality attributes, so you may opt for merging the two sections.

High-level architecture

Start with some information about Components, Activities and Infrastructure (respectively, use UML Component, Activity and Deployment diagrams). Then go into detail about each component and link to different markdown files.

Ensure that only components that have actually been implemented appear in this section, and that each is accompanied by a small description of its scope. There should be only one team identified as owning each component (e.g., perhaps the two teams currently identified as owning Calendar are actually owning two distinct components related to calendaring?). Also, each team should own at least one component (e.g. T2G3 is not currently listed as the owner of any component).

This section should also include any information that does not fit a specific component but pertains, nonetheless, to design/architecture or to the use of any of the patterns that we have studied during the semester or applied in the project (e.g., monitoring, could patterns, devops, etc.).

Operation

This section is missing instructions about:
 * how to set up a production environment,
 * how to build and package the system for deployment,
 * how to deploy the system to production,

It is not clear if the “How to run” section refers to development or production environments (or to both). If this information does refer to the production environment, then it does not seem to be sufficient. In practice, how do we deploy and run the system?

This section also misses to state explicitly if any architectural fitness functions were implemented, how they relate to the identified quality attributes, and how they are actually measured.

Usage

It should be clearer whether there are two APIs (*internal* and *external*) or just one. This section should be mostly about the external API, with the internal API being described in the “Contributing” section.

Authentication.md (T2G1)

Besides the comments provided specifically about this component, consider carefully also the ones about the other componentes, and adapt them to your own context.

Endpoints

All information stated in this section should already be part of the Swagger-based API documentation. Therefore, instead of describing each endpoint in the report’s text, please link to the API documentation: https://uni4all.servehttp.com/api-docs/#/Authentication and https://uni4all.servehttp.com/api-docs/#/User
Furthermore, I suggest you keep mostly within the structure of the general report, and rename this section to “Usage”.

Technologies

This section should be about the technologies and tools used in the component, but also include the rationale for choosing them. You are missing to explain why these technologies were chosen. For example, why redis instead of postgres to store sessions?

Design and architecture

Start this section with any relevant information about **(Sub-)Components**, **Activities** and **Infrastructure** (respectively, use UML Component, Activity and Deployment diagrams) of this specific component. Be sure to state clearly relationships that may exist between this and other components.

- Perhaps the higher-level architecture of the system will be easier to identify after you consider our feedback in section *“High-level architecture”* above, but I don’t think it is true when you write that *“Our application makes use of a Microservices architecture”*. In any case, an *Access Token* is not useful only in the context of microservices.

- When describing each pattern instance, please point to the reference(s) that you used for that pattern (POSA, Enterprise, API Patterns, CAP, other).
- Some aspects of the solution are not clear. The diagram does not seem to show key elements of the implementation (e.g., redis, postgres) and it uses a notation that does not seem to be UML and, therefore, requires explanation: does the arrow represent data flow or dependency? Why are the connections with the several services not arrows? Additionally, the diagram gives the impression that all requests go through the "Authentication Service", who decides where to forward the request, is that right?
- When stating the consequences, do so explaining *why* that consequence exists. For example, instead of just writing that there is more *flexibility*, explain also exactly where that flexibility comes from.
- In the consequences you state that *"access tokens are not stored on the server"*. I had the impression that they were stored in redis, is this not true?

Contributing

The section "Scraping of Sigarra's protected pages" is apparently mostly about the *scraping* component, rather than about the authentication component.

Scraping.md (T1G4 , T1G3 - meals)

Besides the comments provided specifically about this component, consider carefully also the ones about the other componentes, and adapt them to your own context.

I suggest you keep mostly within the structure of the general report, and rename the "Flow Diagrams" section to something like "Design and architecture".

The *Mapping* sections should explain the solution of the pattern and how each element of **this** implementation maps to that solution. For example, the mediator pattern defines a few roles and it would be worth stating clearly what elements of your implementation are playing those roles.

When stating the consequences, do so explaining why that consequence exists, and relating that to your specific implementation when applicable. For example, why exactly do we get to *"reuse individual components more easily"*?

Chat.md (T1G3)

Besides the comments provided specifically about this component, consider carefully also the ones about the other componentes, and adapt them to your own context.

High-level Architecture

Please see specifically the comments to *t1g2-calendar.md* regarding the diagram, as they are applicable to the diagram in this section too.

Technologies

Mongodb is presented as having benefits only, which likely missed to tell the full story. What are you abdicating when choosing mongodb?

Design and Architecture

You state that *"the chat backend consists of 3 microservices"*. I wonder to what extent these are really microservices, as they don't seem to comply with some of the characteristics of microservices that we have studied in classes (e.g., Organized around Business Capabilities, Decentralized Governance, etc.).

This component has some *design and architecture* documentation in this section, but it's not oriented to the structure suggested in the original template for the report. You should focus here in describing the main solutions that have been employed. Ideally, such solutions will be patterns, but you can describe any kind of solution, even if they have not been described in any of the pattern catalogs that we have studied.

There are quite a few API patterns documented for this component, which is great, but you should document them using the same structure used in the *Design and Architecture* section (i.e., Context, Mapping, Consequences).

Payments.md (T1G1)

Besides the comments provided specifically about this component, consider carefully also the ones about the other componentes, and adapt them to your own context.

The documentation for this component should explain why these technologies were chosen.

This component is lacking considerably in *design and architecture* documentation. Please consider the "Design and architecture" section in the original template, as it should help to describe the main solutions that have been employed. Ideally, such solutions will be patterns, but you can describe any kind of solution, even if they have not been described in any of the pattern catalogs that we have studied.

t1g2-calendar.md (T1G2)

Besides the comments provided specifically about this component, consider carefully also the ones about the other componentes, and adapt them to your own context.

The diagrams seem to be UML component diagrams, but there are a few notation details that do not seem correct. For example:

- *Dependencies* are the only kind of arrowed association in component diagrams, but they are represented using a dashed line.
- Some of these arrows seem to be representing data flow (for example: the three associations between *SIGARRA* and *Scraping*), although that is not something that can be represented in a component diagram. For representing behavioral aspects of the system, please consider using activity diagrams, for example.

The *Mapping* sections should explain the solution of the pattern and how each element of **this** implementation maps to that solution. The diagram used to document the use of the *Results Cache* pattern is useful for this but it doesn't tell the whole story.

There are quite a few API patterns documented for this component, which is great, but you should document them using the same structure used in the *Design and Architecture* section (i.e., Context, Mapping, Consequences).

t2g4-calendar.md (T2G4)

Besides the comments provided specifically about this component, consider carefully also the ones about the other componentes, and adapt them to your own context.

This component is lacking considerably in *design and architecture* documentation. Please consider the "Design and architecture" section in the original template, as it should help to describe the main solutions that have been employed. Ideally, such solutions will be patterns, but you can describe any kind of solution, even if they have not been described in any of the pattern catalogs that we have studied.

groupMaking.md (T2G5)

Besides the comments provided specifically about this component, consider carefully also the ones about the other componentes, and adapt them to your own context.

The instantiation of the "API Description" pattern in the Uni4all project goes well beyond this specific component. Perhaps this would fit better in the "general" part of the report instead?

The context sections should be about the aspects of the system (or of the development process) that justify the selection of the pattern. For example, for the "API Description" pattern, it

would also be relevant to explain the importance of other developers understanding the API, so that they can use the module.
When stating the consequences, do so explaining why that consequence exists. For example, instead of just writing that the description is easy to evolve and maintain, explain also what exactly makes it simpler that other approaches.

feedback.md (T2G2)

Besides the comments provided specifically about this component, consider carefully also the ones about the other componentes, and adapt them to your own context.

This component is lacking considerably in *design and architecture* documentation. Please consider the “Design and architecture” section in the original template, as it should help to describe the main solutions that have been employed. Ideally, such solutions will be patterns, but you can describe any kind of solution, even if they have not been described in any of the pattern catalogs that we have studied.

jobs.md (T1G2)

Besides the comments provided specifically about this component, consider carefully also the ones about the other componentes, and adapt them to your own context.

This component is lacking considerably in *design and architecture* documentation. Please consider the “Design and architecture” section in the original template, as it should help to describe the main solutions that have been employed. Ideally, such solutions will be patterns, but you can describe any kind of solution, even if they have not been described in any of the pattern catalogs that we have studied.