Engenharia de Serviços (MEI / MES)

2024/2025

Practical Project, Part 2

Applying service design techniques to model a real-world service

Deadline #1 (part 1, for feedback): 14 March 2025 (23h59m)
Deadline #2 (part 1+ part 2, for assessment): 23 May 2025 (23h59m)
Submission via Inforestudante

Note: Academic fraud is a serious ethical breach and is not admissible behavior for a student and future practitioner. Any attempt of fraud may lead to the cheater and their accomplices failing the course. Other sanctions may additionally apply.

Objectives

Apply the service design techniques to model a service, and, in doing so:

- Gain an understanding of the complexity of services and the need for the said techniques.
- Develop competencies in using those techniques for diagnosing and evolving existing services and for designing new ones.
- Apply the cloud-based technologies you have learned to create a simple, powerful, responsive, well-designed, and beautiful service.

Submission

For part #1 (specification of the service) of this assignment, you must submit:

- Persona(s) – two well-defined personas are enough for the purpose of the assignment.
- Customer journey map(s).
- Stakeholder map(s).
- Expectation map(s).
- Other elements that the groups deem relevant.

The first three instruments are available in the Smaply software used in the course. Others require additional forms or tools. Further details are provided in class.

A set of PDFs with the deliverables must be generated and submitted via Inforestudante by the deadlines.

For part #2 (revised specification + implementation) of this assignment, you must:

- Develop part of the assignment on the Amazon AWS cloud. This means that you should install and configure multiple services on the Amazon AWS cloud.

Pólo II · Pinhal de Marrocos · 3030-290 Coimbra · Portugal · Tel.: +351 239 290 000 · info@dei.uc.pt · http://www.uc.pt/fctuc/dei

- Keep your installation on Amazon until you present the assignment. Keep it untouched and remember that many services keep track of the dates they were updated.
- Make sure that you keep a comfortable margin from your budget to present the assignment in the final defense.
- You should bundle all the code and other elements that you do not keep online on Amazon and deliver them in Inforestudante before deadline #2. Please keep your file sizes as small as possible, by uploading only the source code. Do not upload compiled and public software libraries that you have running with the code. You do not need to deliver a report.
- By deadline #2, you must also re-submit an improved specification of the service based on the feedback you received in part #1. You must **include a brief document clearly identifying all the changes** you made to your original specification to address the feedback.

IMPORTANT: Before starting each part of the project development (i.e., design and development), the group must obtain the professor's approval regarding the distribution of tasks among its members. Regarding the split of work, students are required to showcase their acquired skills in both the design and implementation components of the project. It is essential to ensure that all students engage in both aspects of the project, rather than dividing tasks in a way that some students are solely responsible for design while others handle implementation. This approach will considerably penalize the entire group. This aspect is assessed during project development as well as in the final defense.

Overview

This project aims to develop and implement a modern computer repair service for PrimeTech Repairs. Customers begin by visiting the PrimeTech Repairs website, where they can browse predefined repair services such as screen replacement, battery replacement, software troubleshooting, and virus removal. Each service includes a price estimate based on standard repair costs. Customers then select a service and book an appointment, choosing between standard or urgent repair, with urgent repairs incurring a non-refundable urgency fee. In case they already have an account, they can log in and associate this request. They can leverage face recognition to authenticate.

At the scheduled time, customers bring their devices to the shop, where a technician performs a brief inspection to confirm that the selected service applies. Customers then pay a non-refundable diagnostic fee, which will be deducted from the final bill. If an urgent repair was selected, the urgency fee is also paid at this stage, but it is not deducted from the final cost.

A detailed diagnosis follows, and if additional repairs or higher costs are required, the shop contacts the customer to explain the situation and request approval before proceeding. Approval is done through the website after login, where a detailed description of the additional work and costs is available. If the customer agrees, they select the

Pólo II · Pinhal de Marrocos · 3030-290 Coimbra · Portugal · Tel.: +351 239 290 000 · info@dei.uc.pt · http://www.uc.pt/fctuc/dei

appropriate option online, and the repair continues with the updated price. If they decline, they select the appropriate option and can retrieve the device without any further work beyond the initial diagnostic.

Once the repair is complete, the customer is notified via SMS or email. They then pick up the device and pay the remaining balance, which is the final repair cost minus the previously paid diagnostic fee. An invoice is sent via email.

Throughout the process, customers can track the repair status through their online account.

References

Researching facts and not making assumptions is part of the process of good service diagnosis and design. Feel free to investigate real services online. The instructors are also available to discuss your options. Here are also some examples:

- Micro Center - https://www.microcenter.com/site/service/instore-service-diagnostics.aspx
- NerdsToGo - https://www.nerdstogo.com/on-demand-it-services/

Important aspects (based on errors frequently made by students)

Regarding personas
The descriptions of personas must be rich and detailed. They must be credible as if we were describing real people. Only knowing people well enables you to design a service that suits them. Regarding the number of personas, it is not about being a lot or just a few, but how different and complete are the described profiles and needs. For instance, it does not contribute a lot to the service design if we have a lot of personas with basically the same needs; but we should not leave out important profiles.

Regarding customer journey maps
Being so rich, this is one of the most important tools in service design. It enables us to understand how the customer "travels" along our service. It is almost like a movie, where we have various scenes or snapshots in sequence. One of the most important aspects—see slides and book—is to make sure that we have the most adequate touchpoints (the moments of interaction). Journey maps are also very powerful in the sense that they enable us to relate what the customer sees and does with back-office actions and systems and the channels that are used for the interaction in touchpoints. If the customer receives a notification by SMS (channel), then there must have been a back-office system/person/process sending that message (back-office lane)—all these events and lanes must be consistent with each other. It is the proper synchronization of people, technology, and processes that ensures that the service flows smoothly. Pay close attention to how front-end systems and back-end systems interact across various channels. All must be consistent in the customer journey map. Remember that a channel is a "means

Pólo II · Pinhal de Marrocos · 3030-290 Coimbra · Portugal · Tel.: +351 239 290 000 · info@dei.uc.pt · http://www.uc.pt/fctuc/dei

for contact": email, phone, SMS, face-to-face encounter, land mail, etc. Product or money are not channels.

Regarding the number of maps, check the slides and book. It all depends on the level of abstraction and detail that you decide is adequate. You may have "happy path" scenarios, exception scenarios, different maps for different ways to use the service, etc. Please also remember that your maps must be understandable. Avoid too much clutter in one map (e.g., lots of personas).

It is frequent for people to forget touchpoints when modeling. Remember that confirmation emails/SMS are touchpoints, email/SMS warnings of the impending arrival of the order at your home are touchpoints, and the physical interaction with the delivery person is a touchpoint.

Regarding stakeholder maps
It is key to identify the different importance of the various involved stakeholders. Keep things clear, so that someone else can understand the exchanges between the various actors. The number of maps to create depends on the different scenarios of exchanges that you want to explain.

Regarding expectation maps
Expectation maps should be consistent with the profiles and needs of your personas. It does not make sense to have several different personas, with different motivations, and then just the same expectation map for all of them. Indeed, some expectations may be common, but others will be different. For instance, someone with a lot of money and little time has different expectations than someone short on cash. The expectations of a young active person are different from those of a senior or handicapped person.

Implementation

Students should implement the system as depicted in Figure 1, which uses multiple AWS services. Students should write the frontend in JavaScript, preferably with a framework like React or Svelte.

Pólo II · Pinhal de Marrocos · 3030-290 Coimbra · Portugal · Tel.: +351 239 290 000 · info@dei.uc.pt · http://www.uc.pt/fctuc/dei

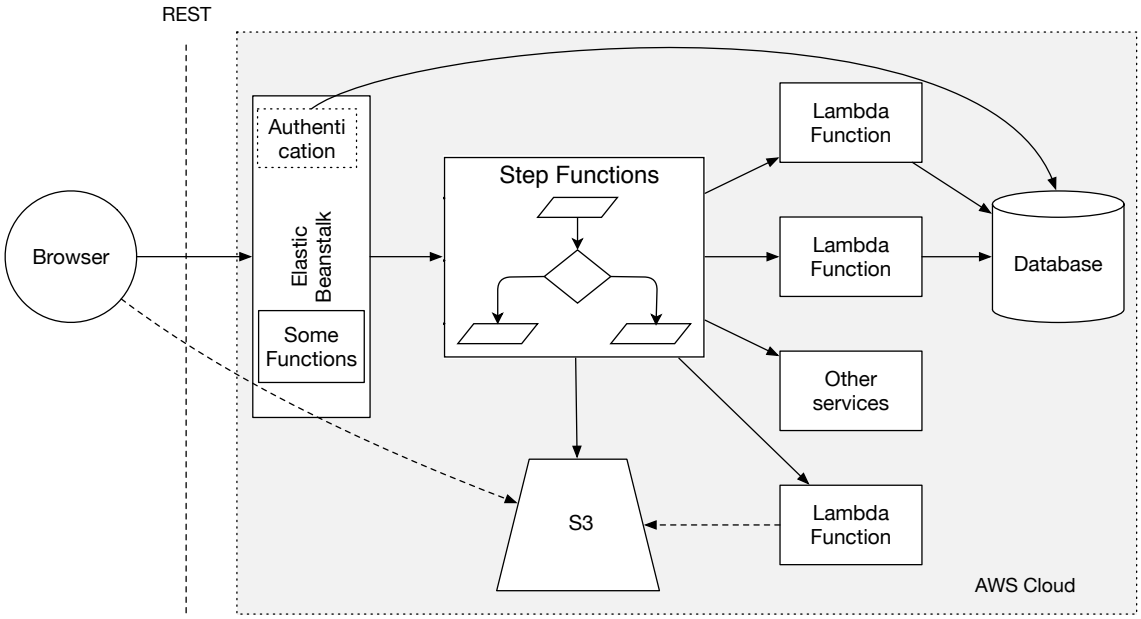

Figure 1 - Container Diagram of the Project

The frontend applications will access the backend to perform, at least, the following operations:

- Log in using facial recognition/Log out.
- Get information from the shop.
- Submit a repair request, including an appointment by selecting a date.
- Pay.
- View the status of requests.

The frontend applications interact with the backend using a REST API supported by a Django project. Students are encouraged to use the Django REST framework for building APIs, as it offers more tools and features compared to the vanilla Django. This Django layer should be minimalistic and is essentially a gateway for other services. Even though students may implement some functionality in the Django server, this is not a Django assignment! The Django technological stack is not mandatory; students may use other technologies, but, in this case, they should discuss their options with the professor. Students should deploy the Django server (or preferred alternative) in **Elastic Beanstalk**.

One Step Functions Workflow should serve as the backbone of the backend logic. Workflows have the great advantage of being easier to understand than Python code. This makes the interaction between data scientists, engineers, and managers running a software project simpler. In the case of this assignment, they may control timeouts on customer responses. In this application, the workflow may need to interact with different services, including **Lambda** functions and databases, like **Dynamo**. Lambda functions will usually provide additional capabilities to the, otherwise limited, workflow. **Django should not need to directly call Lambda functions**.

Users start by visiting the web page of PrimeTech Repairs, where they can view the options available, prices, schedule options, etc. Before proceeding to set a repair operation, they need to log in to the service using facial recognition (users may also log

Pólo II · Pinhal de Marrocos · 3030-290 Coimbra · Portugal · Tel.: +351 239 290 000 · info@dei.uc.pt · http://www.uc.pt/fctuc/dei

in before doing any search). At some point, once the user decides what to do and presses a service request button, a (Step Functions) workflow starts on the server. This workflow will control the process as follows:

- It will end, if the user does not show up at the appointed date.
- After going to the repair shop, if the user decides to proceed with the reparation, it must allow wait for a certain time for the online payment. Otherwise, the item must be sent to the "lost" section of the shop.
- It must finish once the user gets the item back from the shop. Again, the user has a limited amount of time to recover the item.

Students should handle a few additional concerns:

- The application must manage the schedule of visits to the shop to prevent overlapping appointments.
- Customers must be able to check the status of the repair (scheduled, waiting for payment, or finished).
- The payment operations should be idempotent. Even though the application does not need to keep track of bank accounts or balances, it should ensure that only one transfer order happens.

Students should implement a second frontend application for the shop management, where they list appointments, as well as current and past services. A button to set the repair as finished should be available. This second website does not require a login.

Sending an SMS, an email, or some other notification is never necessary in the implementation.

Utilization of Technologies

Authentication/authorization libraries
Students should implement their own authorization and authentication mechanisms with JSON Web Tokens. This enforcement serves tutorial purposes alone, because rewriting security sensitive code, instead of resorting to standard libraries, could lead to major security failures.

Boto3
Boto3[1] is a Python Software Development Kit (SDK) to operate AWS resources. The Django project and the Lambda functions will make extensive use of Boto3. Boto3 includes APIs to control a large spectrum of AWS services, e.g., databases, or Step Functions.

Storage
Students should minimize the use of the database that is configured by default in a Django project, SQLite (preferably not use it at all), and use RDS and other AWS services, like SimpleDB or DynamoDB. Again, for training reasons, usage and configuration of **more**

[1] https://boto3.amazonaws.com/v1/documentation/api/latest/index.html.

Pólo II · Pinhal de Marrocos · 3030-290 Coimbra · Portugal · Tel.: +351 239 290 000 · info@dei.uc.pt · http://www.uc.pt/fctuc/dei

than a single type of databases is encouraged, e.g., to keep authentication data for the log in and to keep other information.

Storage of HTML and JavaScript
Even though a good solution to store HTML and JavaScript would be AWS S3, this option will raise several challenges related to Cross-origin resource sharing (CORS). Hence, students may prefer to serve these contents directly from their Django project.

Rekognition
According to AWS[2], "Rekognition offers Computer Vision capabilities, to extract information and insights from images and videos". To set Rekognition ready to identify persons in photographs, students need to perform a preliminary step of preparing a collection of faces and the creation of an external index to match the faces that Rekognition identifies with external real users. To improve results students may use more than one picture per person.

Additional Challenge

A challenge for students is to define their interface using Open API/Swagger. In addition to providing a clear definition of the REST interface, this technology enables the creation of fancy testing interfaces, server skeletons, and other valuable features.

[2] https://aws.amazon.com/rekognition.

Pólo II · Pinhal de Marrocos · 3030-290 Coimbra · Portugal · Tel.: +351 239 290 000 · info@dei.uc.pt · http://www.uc.pt/fctuc/dei