Engenharia de Serviços / Service Engineering (MECD)

2024/2025

Practical Project, part 2

Applying service design techniques to model a real-world service

Deadline #1 (part 1, for feedback): 18 October 2024 (23h59m)

Deadline #2 (part 1+ part 2, for assessment): 20 December 2024 (23h59m)

Submission via Inforestudante

Note: Academic fraud is serious ethical breach and is not admissible behavior for a student and future practitioner. Any attempt of fraud may lead to the cheater and its 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 competences 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.

Final Delivery

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

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

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

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 deadline.

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

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

Overview

The goal of this project is to model and implement the service of requesting a bank loan online.

The customer begins by accessing the bank's website, specifically the loans page, where they can use a simulator to set the loan amount, duration (maturity), or monthly payment. After finalizing these details, the customer must log in to the bank to complete the loan application, requiring authentication via facial recognition. Based on the selected loan type, the system prompts the customer to enter the necessary information (e.g., monthly income, regular expenses) and/or upload required documents (e.g., salary slips). The system processes this data to calculate a credit score (e.g., using a machine learning module) and classifies the application into one of three categories: (i) accept, (ii) interview, or (iii) reject. A loan officer is then assigned to the request, receiving the credit score information to decide whether to approve, reject, or request an interview. The system records the decision and notifies the customer via SMS or email, depending

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

on their preferences, that a decision is available in the application. If an interview is required, the loan officer provides available time slots, and the customer can select the one that best suits them. Following the interview, the loan officer makes a final decision, records it in the system, and the customer is informed via SMS or email that the decision is available in the application.

Bank employees access the system through conventional login and password procedures, granting them access to a dashboard with statuses for the different loan request processes.

You may assume that accounts were previously created (i.e., they already exist).

References

Researching facts and not making assumptions is part of the process of good service diagnosis and design. Feel free to investigate real services like the one described for inspiration in modeling yours (access banks' websites to explore additional information and ideas). Look also into software that supports online loan processes, such as Digital Loan Origination System (https://www.inlaks.com/our-business/inlaks-digital-solutions/marketplace/digital-loan-origination-system/), Loan Management Software (https://www.creditonline.eu/), or similar tools.

The instructors are available to discuss your options.

Important aspects (based on errors frequently made by students)

Regarding personas

It is important that the descriptions of the personas are 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's not really 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's 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

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

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 backoffice system/person/process sending that message (backoffice lane) — all these events and lanes must be consistent with each other. It is the proper syncronization 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 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, 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.

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

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.

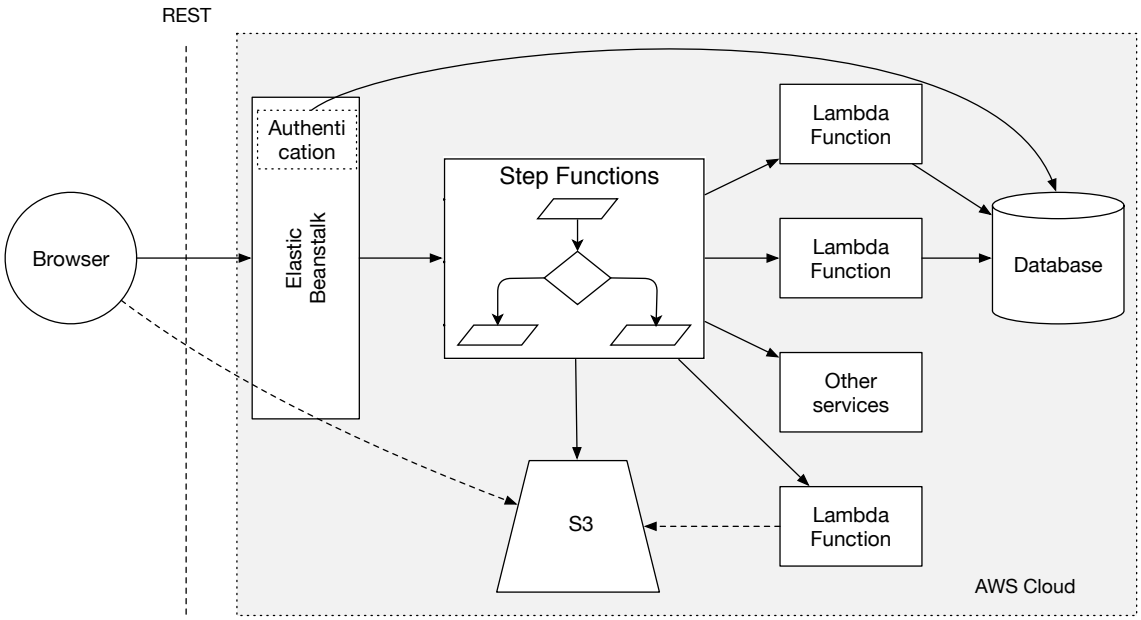

Figure 1 - Container Diagram of the Project

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

- Log in/Log out.
- Get information from the bank.
- Access the loan request page.
- Submit a loan request.
- View the status of the request.

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 vanilla Django. This Django layer should be minimalistic and is essentially a gateway to other services. Importantly, this is not a Django assignment! Students should not write all the logic in the Django project. Some of this logic should be ensured by a Step Functions Workflow. They should implement their own solution using JSON Web Tokens. Students should notice that this enforcement serves tutorial purposes alone. In a real scenario, they should resort to standard

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

libraries. Access to the backend goes through **Elastic Beanstalk** running the Django server. The Django technological stack is not mandatory; students may use other technologies, but, in this case, they should discuss their options with the professor.

A Workflow written in **Step Functions** should be a central part of the backend implementation. Workflows have the great advantage of being easier to understand than Python code. This makes the interaction between data scientists, engineers, and managers simpler. In the case of this assignment, they may control timeouts on bank responses. In the process, the workflow may need to interact with different services, including **Lambda** functions and databases, like **Dynamo**. Lambda functions will usually provide support to the workflow. Django should not need to directly call them.

Users start by visiting the web page with the loan simulator and fill in some numbers and fill in some data. Once all data is ready, they will have the option to press a button to proceed with the simulation. At that point, they will have to log in using facial recognition, supported by AWS Rekognition. The bank will then provide additional requests for data, such as a pdf with the job earnings. When all data is ready, the user presses another button that will start the workflow, running in Step Functions. This workflow will apply the rules to (i) accept, (ii) interview, or (iii) reject the loan (at this point, an existing credit score algorithm may be used). The workflow also serves to keep the state of the interaction between the client and the bank. The workflow should send a notification to the loan managers[1], letting them know about the submission and, in some cases, the need for setting up an interview. It should also help the bank ensure that clients get an answer before some deadline. Students may also resort to a database to keep some information, e.g., the existing loan requests, possibly by the same client.

Students should do a second frontend application for the bank management side, where they list current and past loan requests and where bank workers can manage such requests, e.g., by setting up a date for the interview and deciding the result of the loan. If an interview is required, the loan manager should provide a selection of available time slots, allowing users to choose the one that best suits their convenience.

[1] No real notification is needed.

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

Utilization of Technologies

Authentication/authorization libraries

Students must not use the Django (or whatever framework they use) **authentication and authorization** libraries. Implementing our own version of security-related libraries is usually a very bad idea. We are doing this for the sake of training.

Boto3

Boto3[2] 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 than a single type of databases is encouraged**, e.g., to keep authentication data for the log in and to keep information of the state of a ticket.

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[3], "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.

[2] https://boto3.amazonaws.com/v1/documentation/api/latest/index.html.
[3] https://aws.amazon.com/rekognition/?nc1=h_ls.

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

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.

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