Engenharia de Serviços (MEI / MES)

2021/2022

Practical Project

Applying service design techniques to model a real-world service

Deadline #1 (part 1, for feedback): 25 March 2022 (23h59m)
Deadline #2 (part 1+ part 2, for assessment): 21 May 2022 (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.

Submission

For the 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 maps(as).
- 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.

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 the 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.
- 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 stating the changes that you made to your original specification to address the feedback.

Overview

The goal of this assignment is to model and implement a modern restaurant service. When entering the restaurant, the user starts by picking up a location tag and then proceeds to choose the meal on a big touchscreen where all the options are presented. After composing and submitting the order and indicating the number of the location tag, the user must pay, using the restaurant's face recognition technology. Upon successful payment, the restaurant sends the user an email with the receipt. The user can then find a table to sit. When the meal is ready, a robot will use the location tag to find the user and deliver the food. A final face recognition process is used to ensure that the recipient is the correct one.
You may assume that the payment account has been previously topped up.
When the kitchen staff authenticates in the system using traditional login and password, they can see the pending orders and their respective status.
Note that the kitchen staff is not the "customer". Consider where the above information about it should be used in the diagrams that you need to submit.

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. The instructors are available to discuss your options.

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

Left: locator tag; Right: Order touchscreen

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 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 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,

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

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 application as depicted in Figure 1, which uses multiple AWS services. Students should write the frontend application to run on the browser using React. The React application interacts with the backend application using a REST API supported by a Django project.

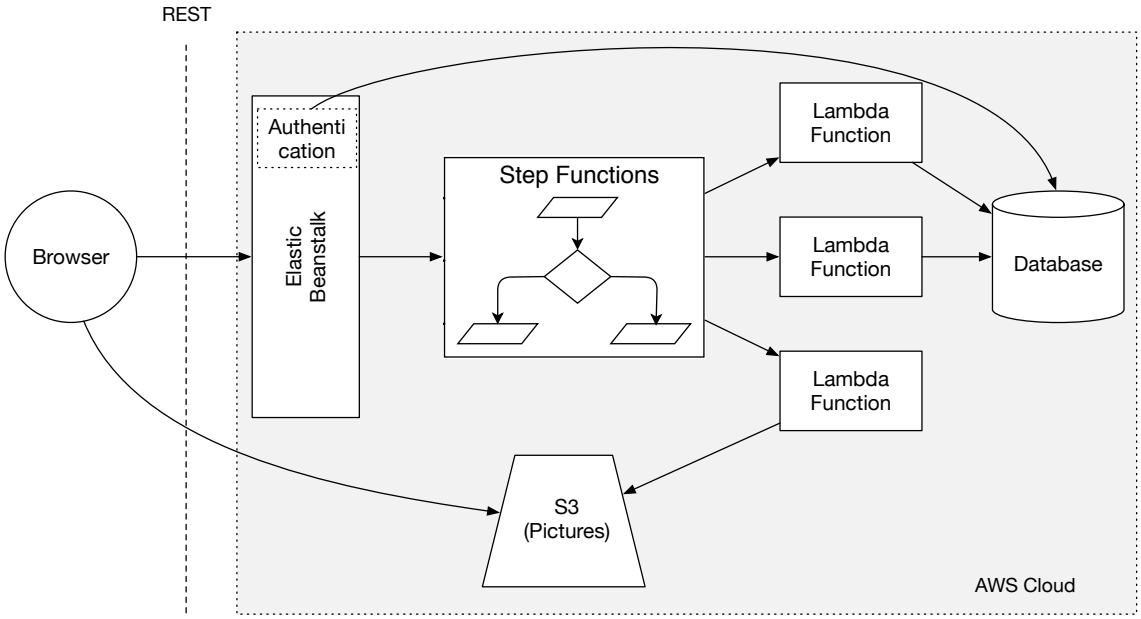

Figure 1 - Container Diagram of the Project

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

- Get the list of food items to create a meal.
- Submit a meal order including the location tag number.
- Take a photo of the user to do the payment.
- Log in/Log out (only for the kitchen staff).
- List orders and status (only for the kitchen staff).
- Confirm meal delivery.

Access to the backend goes through **Elastic Beanstalk**. Students may use standard **Django** (as opposed to the Django REST framework) in Elastic Beanstalk. This 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. They are encouraged to create good workflows in Step Functions. Also, students must not use the **authentication/authorization** libraries available in Django. They should implement their own solution using JSON Web Tokens. This technological stack is not mandatory; students may use other technologies, but, in this case, they should discuss their options with the professor.

The **Step Functions** component is the heart of the project. It keeps track of workflows, e.g., processing a meal order to determine the operations to do at each step. In the process,

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 workflow may need to interact with **Lambda** functions, or it may need to invoke Rekognition.

To provide a list of foods that are available, students should provide a simple web service using the Django project. This service should reply with a list of food items that the user composes in the web application, like soup, salad, chicken sandwich and others. Once the user chooses what to buy, the program should compute the price of the meal (on the server side), show the price to the user, and let the user take a picture for the face recognition payment. The frontend must also send the number of the location tag to the backend. The server does not need to keep track of the food items that the user intends to order while the client is taking his or her options. The client side should keep all the state. At this point, if the user presses a button to complete the purchase, the server starts a workflow in step functions. The workflow will keep all the interaction state from this point on.

The workflow receives the previously selected food items and a picture of the user (although a pointer to the picture is likely a better idea). Then, it uses Rekognition to find the user corresponding to the picture. If Rekognition cannot find any user, the display should show an error message telling the client to go to the customer support. Otherwise, if the user is found, the workflow passes the order to the kitchen and waits for the meal to be complete. Students may simulate this with a wait period in the workflow. When the meal is ready, a robot is dispatched with the food to the location of the tag. When the robot arrives, the robot confirms the delivery, by taking another picture of the user, to confirm the user's identity using, once more, face recognition. Students may simulate this final step with a button in a web application interface. They do not need to repeat the process of taking a picture of the user with the browser.

Students need to provide two separate front-end applications: one for the regular users purchasing a meal and another one for the kitchen staff (no sign in is necessary, only log in and log out). Students should write both applications using React.

Utilization of Technologies

Regarding Rekognition

According to AWS[1], "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.

Regarding 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

[1] https://aws.amazon.com/rekognition/?nc1=h_ls.
[2] 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

includes APIs to control a large spectrum of AWS services, e.g., databases, or Step Functions, just to provide a couple of examples.

Regarding Step Functions
In this project we will take advantage of workflows to keep application state. The most complex parts of the workflow usually involve interaction with real users, like waiting for the meal to be cooked or for the user to confirm the reception of the food. We suggest a couple of possibilities that students may discuss and explore. One such solution is the use of activities. Activities enable the workflow to wait for a human to complete one task. The workflow can start an activity for a third party to pick at any point. Another possibility that students may follow involves the use of tags and messages. A running workflow may add tags to the workflow state to let the cookers get information about this order. Cookers may respond in the end by sending a message. These tags might also serve to let the browser application determine if it is ready to display the final confirmation button. Students should discuss other options with the professors.

Regarding Storage
Students may use the AWS Simple Storage Service, S3, to keep photographs. They should minimize the use of the database that is configured by default in a Django project, SQLite, and should use a service in AWS, like SimpleDB, DynamoDB, or even a relational database. Usage and configuration of other databases is encouraged, e.g., to keep authentication data for the kitchen and to keep information of users and their relation to figures indexed by Rekognition. Students are encouraged to use more than a single type of database, like relational and noSQL.

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