Engenharia de Serviços (MEI / MES)

2022/2023

Practical Project, part 2

Applying service design techniques to model a real-world service

Deadline #1 (part 1, for feedback): 10 March 2023 (23h59m)
Deadline #2 (part 1+ part 2, for assessment): 19 May 2023 (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 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

This assignment aims to model and implement a modern pharmacy service for medical prescriptions. It is assumed that customers already have a medical prescription and the respective QR code on their mobile phones.

When entering the pharmacy, the user starts by showing the QR code of the prescription to a reader device at the counter. The list of prescribed drugs is shown on the screen of the pharmacist where he is already logged in with regular username and password credentials. The pharmacist could change each prescribed drug to a generic one if the doctor had allowed it in the prescription. In those cases, the pharmacist must ask customers whether they prefer the generic or not.

After finalizing the list of drugs, the customer must pay using either the pharmacy's face recognition technology, credit card, or cash. Upon successful payment using face recognition, the customer gets an SMS with a number to collect the purchased items and an email with the receipt. In payments with credit card or cash, the customer directly gets a printed receipt with the number that also allows them to collect the purchased items.

Meanwhile, a robot collects the purchased items, and a pharmacy assistant delivers them to the right customer based on the purchase number. After delivery, the assistant updates the purchase status directly in the robot interface.

You may assume that the payment account has been previously topped up. Note that the pharmacist 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 above or the one depicted in Figure 1, for inspiration in modeling yours. The instructors are available to discuss your options.

Figure 1 - Electronic prescription reading at the pharmacy (https://www.digitalhealth.gov.au/healthcare-providers/initiatives-and-programs/electronic-prescribing/for-dispensers)

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 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, 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.

Implementation

Students should implement the system as depicted in Figure 2, 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.

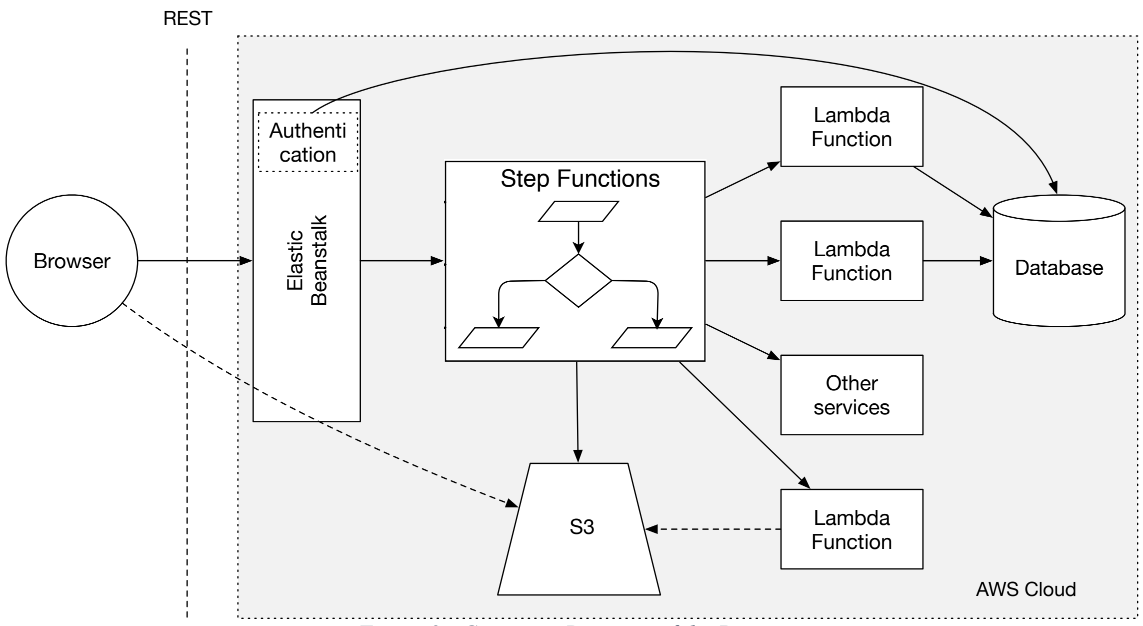

Figure 2 - Container Diagram of the Project

The frontend application will access the backend to perform several operations:

- Log in/Log out.

- List prescribed drugs, price per drug, and alternatives. The server may fill the prescription and additional data randomly, with no need to resort to a database.
- Finalize purchase & pay.
- Provide the delivery state for each purchase already paid.

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

In addition to the backend, the project has other main components:

- The **React frontend** provides the functionality for the pharmacist to log in, read the QR-code[1], get the prescription, select the options for each drug, and process the payment, either manually, with credit card/cash or automatically with visual identification (using AWS Rekognition). Some of these steps may require interaction with the backend, e.g., getting the alternatives for a specific drug or invoking Rekognition. Nonetheless, whenever possible, students should keep state in the client, to ensure that the server is stateless. Students do not need to send an email after the purchase.

- **Step Functions** keep track (i.e., server-side state) of all the process that controls the robot, following the payment. This workflow starts with an activity that conveys a specific task for the robot. Once the robot delivers the medicines it must wait for a human confirmation, before going back to the step of getting another activity.

The pharmacist should be able to check the status of each paid order in the same web interface where he or she processes prescriptions and purchases, e.g., if the robot is picking the items, not started yet, or waiting for delivery confirmation, among others.

Implementation Details

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

[1] Students do not need to use real QR-codes. They may fill URLs with some specific format, such as http://server.com/pharmacy/prescription/1 or they may push a button to go to such URL, thus simulating a prescription.
[2] https://aws.amazon.com/rekognition/?nc1=h_ls.

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.

Boto3

Boto3[3] 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, and use RDS and other AWS services, like SimpleDB or DynamoDB. 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 users and their relation to pictures indexed by Rekognition. Students are encouraged to use more than a single type of database, like relational and noSQL.

Step Function Activities

The interaction between robot and purchases occurs by means of Step Function Activities. Students should understand that the robot has no control over the task it gets. It just runs an infinite loop of requesting a task, completing its job, and getting another task.

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 using their Django project.

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.

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