1 Tell us what pieces of software you think are necessary to develop for the working prototype and how they are related. We call each application (web, mobile or desktop), each API, each batch process that can be deployed independently a piece of software. Support yourself with a diagram if you think necessary.

I propose to our client to create a simple prototype application to test in the first month, and then improve the architecture to allow for better maintenance and scalability, if the prototype works well in the market.

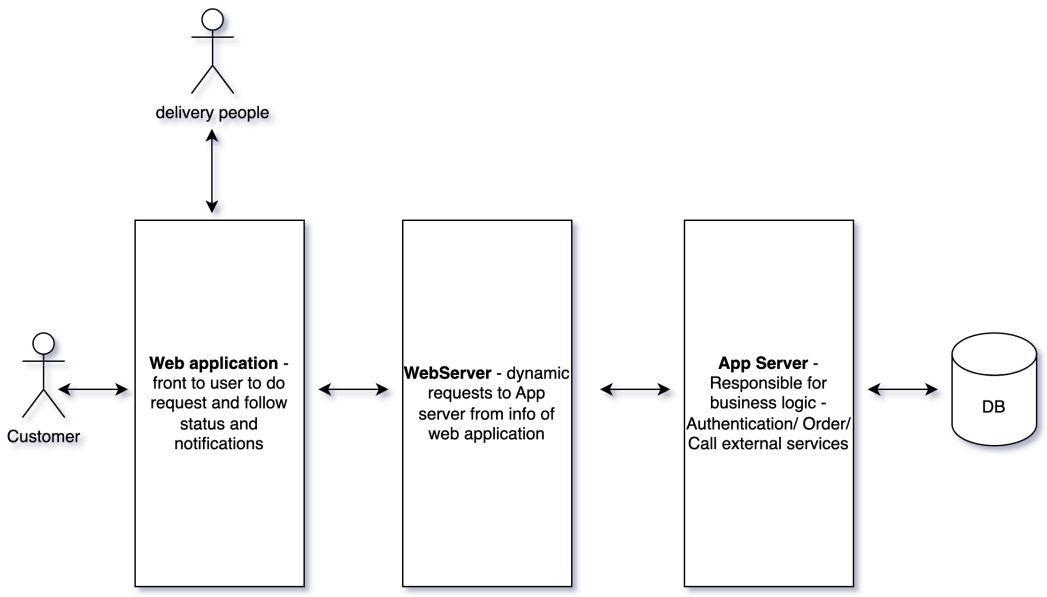

Figure 1: Initial architecuture to prototype of delivery apps.

2 Tell us about the type of architecture you chose for question (1). Monolithic? Micro-services? Any intermediate? Other? Comment on what you based to make this decision.

The type of architecture was monolithic, since we are talking about a prototype (it allows us to develop quickly, focusing on functionality, without the need for a more complex architecture or cross-communication between services).

3 Describe the work methodology you would use for development. It can be some known methodology (Scrum, XP, RUP), an adaptation, or a mixture between several methodologies. Whatever your experience has shown you works. Tell us why you think this form is appropriate for our problem.

For this project, the main points are fast iteration, collaboration and focus on the core application/functionalities (minimum effort, maximum value - MVP).
To start: Design Thinking, the whole team should participate in this part. Having defined the scope and core of our prototype, I used Agile - Kanban and Scrum for development, with quick meetings every day, and a longer one every week, to understand the status of the project. The planning would follow a short Sprint Planning period, and use the longer meetings in the week for retrospective and planning the next sprint.
Since we will be working with prototyping, we can try to create a small prototype in each sprint, and study the feedbacks in our retrospectives.

4 Describe the workflow you would use to collaborate using Git. As with (3), you can use something familiar or an adaptation.

We have a specific repository for the prototype that will be created - *master*.
The *master* is the main one, the ideal is that each employee/team member can participate by creating a branch to develop without changing the main code.
In this branch the developer must do the commit with description of the changes. The change must be reviewed by the team through a code review, where they analyze the quality of the code, scope, if the code is being tested correctly, etc.
The code has been accepted, we can merge and release the change to allow testing of our prototype in Alpha/Beta.
After the good development of the previous phases we can merge into the Master branch.
Important to create canaries to monitor the application. In this case it is a prototype, the idea is to use the user in the "gamma" phase as our canaries.

5 Do you think it is necessary to add any extra member to the team during the development of the prototype? What would your role be? Do you think it would be necessary to add new members after the prototype phase? When and why?

For the prototype phase, I think the team is complete, but to drive development forward, I think we could add a product manager to get feedback from stakeholders on the product definition on each mini-prototype created, gathering requirements and prioritizing features for the prototype. One idea would be to give this responsibility to the impact lead as well, not having to add another person to the project.
After the prototype phase, people will be needed to assist in the development part, for example a QA. The QA will identify and anticipate bugs, ensure functionality and verify that it meets the specified requirements. Test plans and reports are required for good software development and maintenance. Depending on the size of the project and the team it is good to have a project manager, who follows the whole process and can help the communication between the development teams.

6 What other considerations would you have to make the development process robust and efficient?

Robust: Use devops methodology as continuous integration and continuous delivery (CI/CD), helping with maintenance and scalability. To do this, use canaries/alarms and automated testing, to identify problems as early as possible in development, and create KPIs to ensure business goals.
After the prototyping phase, moving to microservices is a good idea, as it is an architecture with greater flexibility and scalability.
Another important point is data security and privacy to gain customer trust.

Efficient: Well defined requirements to avoid risks at launch and refactorings during development.
For project development, use agile methodologies, to drive results and create metrics on progress (using devops KPIs too), improving collaboration and communication between team members and knowledge sharing, encouraging documentation of application architecture, design decisions and key features.