Coding Challenge

Backend API

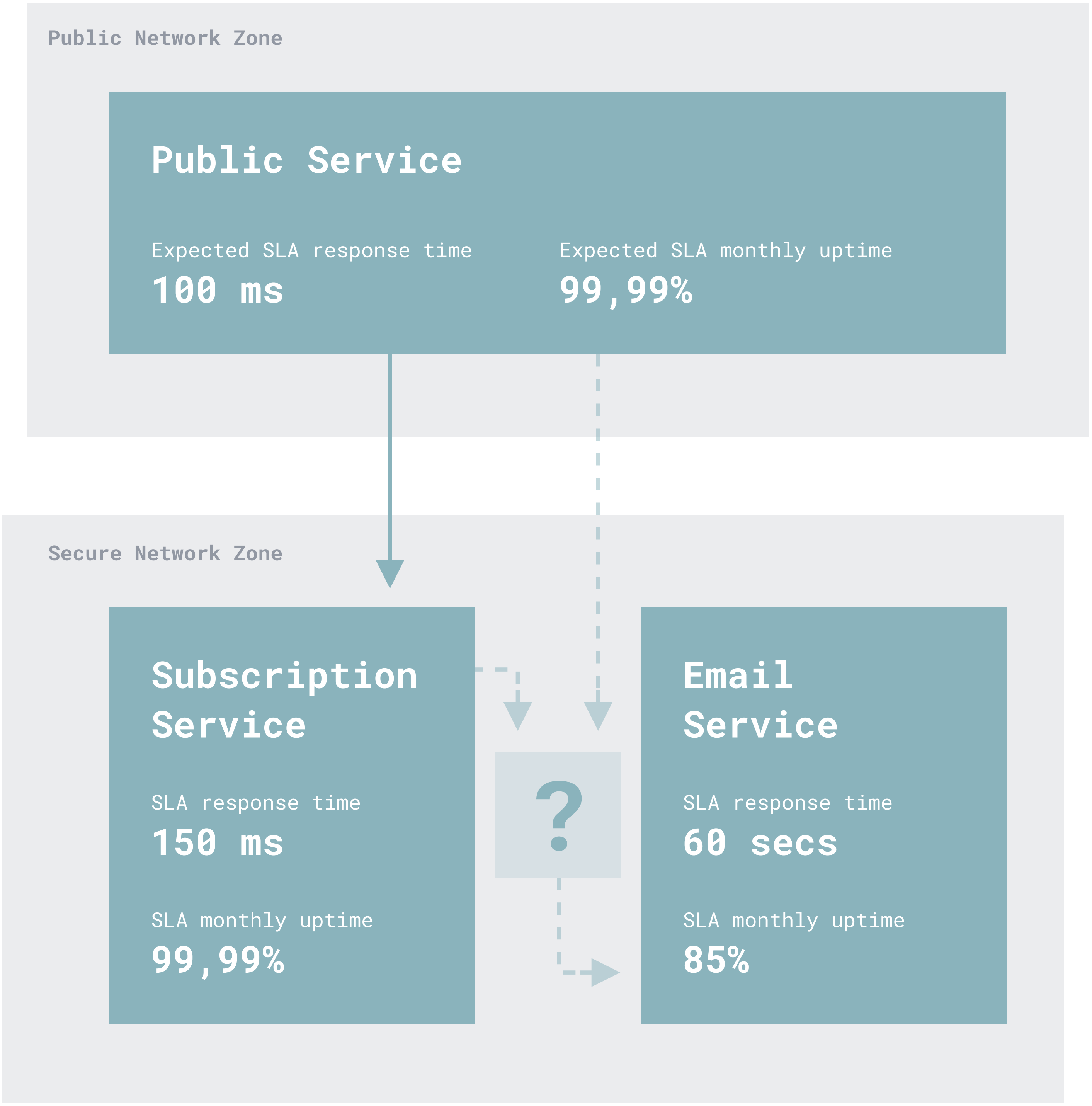
Intro

At iBan, we care about serving consumers (in the sense of applications making use of our data and services) by serving more (high volume) , serving better (low latency) or serving more reliably (high resilience). In general, the topic of High Availability is very important to be ready for the future and deal with the current state with confidence.

In a nutshell, the basic architecture principles are:

- Architect solutions in a way that makes them re-usable.
- Components are designed to scale based on incoming load.
- Always care about data & IT security.
- All solutions should treat core operational database as a costly resource that needs to be used economically.
- Design for Failure – your solution has to assume and handle exception conditions

Architecture and SLA requirements



Mision statement

With the information you have, and some additional assumptions you will have to make, you should develop an API for subscriptions using a framework of your choice (ideally Spring Boot), and Java or Kotlin (we currently favour Kotlin in our development).

Only a single public endpoint is necessary to create subscriptions. Subscriptions contain email, firstName, gender, dateOfBirth, a flag for consent, and the newsletter ID corresponding to the campaign. Only gender and firstName are optional values. The remaining services receive the same parameters.

Services should be secure, only the public service should be accessible from the internet. The subscription service should persist the subscription and return the created subscription to the public service.

To complete the subscription process, once the subscription has been persisted, the email service should receive the necessary information to send an email to the user.

Take into account all the information available, namely the SLAs, when choosing the best approach for each communication between the different services, and other choices you might need to make when implementing your solution.

Development guidelines

Although you are free to choose the technologies you prefer, this is a set of general guidelines that we are expecting to see:

- Create your solution in a public GIT repository (GitHub, Bitbucket, GitLab, etc.), adding the commits as if this was a project you are delivering to a paying client.
- Microservice solution with independent containers for each service.
- Link the services using something like a docker network, no service discovery is necessary.
- Create a README.md explaining how to build/run/use the app, naming every framework/library you use, explain what it does and why you used it.
- Design a solution that satisfies the SLAs.
- Our team should be able to easily check out your code and run it locally, provided we have the most common build tools installed (Ant, Maven, Gradle).
- Provide API documentation using swagger, API Blueprint, Postman or similar.
- Use English as the documentation language.
- **BONUS 1:** Create 1 slide with a CI/CD pipeline proposal for the app.
- **BONUS 2:** Design an architecture to deploy all services in a Kubernetes cluster.

When you’re done send us the link to the repo and any other documentation you want.

Depending on your experience, you might not be able to implement all of these points, and that is ok. If you can’t implement everything, please give us a short explanation of how you think it could be accomplished and what advantages and
disadvantages it has.