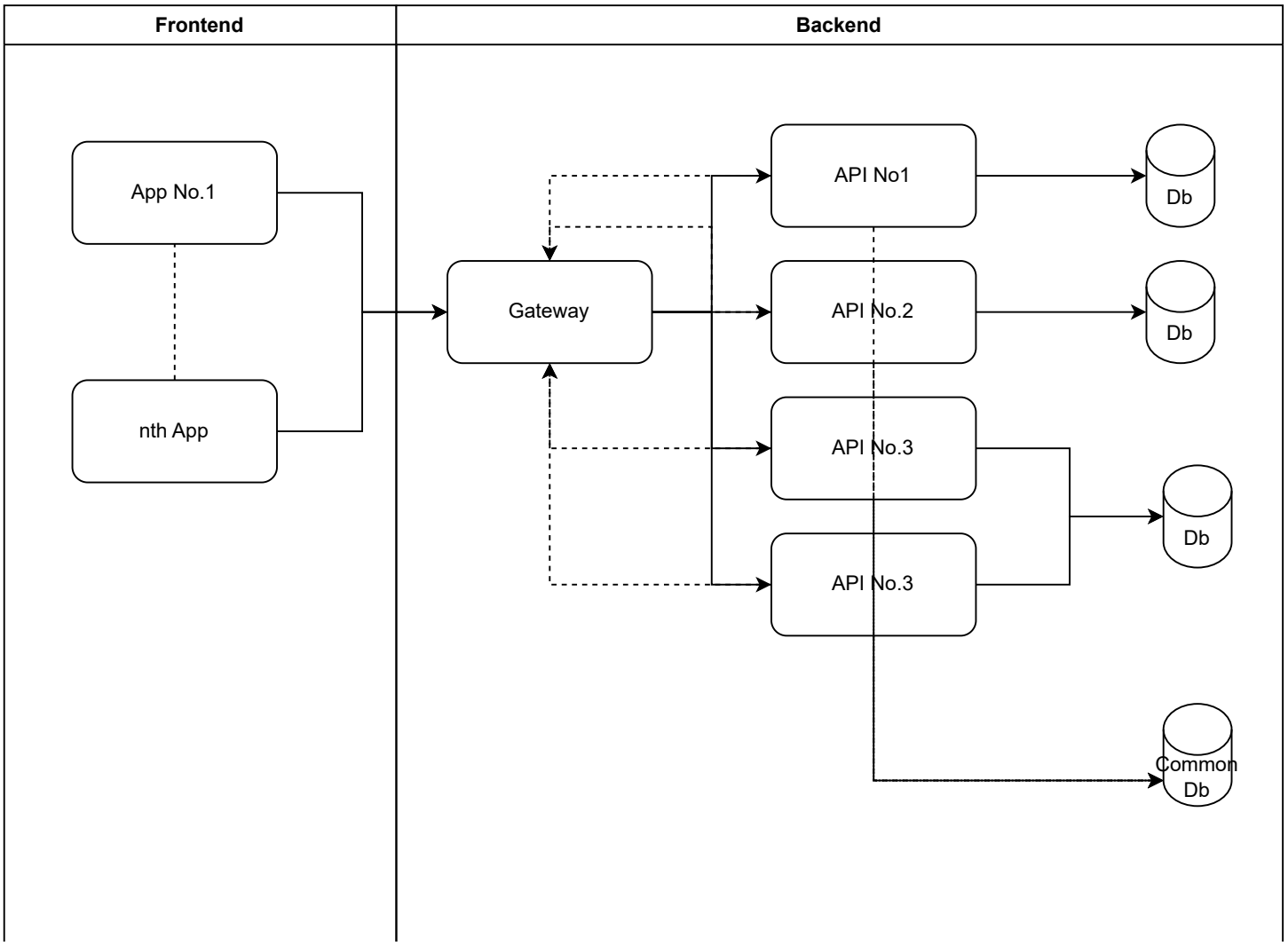
System Architecture

- 10 application will be in communication with gateway api.
- Gateway API will distribute incoming requests to the specified API.
- Every API will have its own database for fault resilience.
- APIs can be deployed more than one so if one API overloads too much gateway with loadbalancer can split that weight between API instances.
- Using docker will significantly reduces the time needed for deploying new API's or increasing current ones.
- Common Db is used for every API to connect, used as Cache Database or Centralized Log Database.
- Every API can have their own log database it can differ according to design choices.
- Services can communicate through gateway so that API doesn't need to know if specific service exist or not. Gateway can be used as orchestrator for backend.
- Again it does differ services can send request between each other it is a matter of design choice
- API doesn't have any connection to outside for security reasons.
- Authentication and Authorization can be used in gateway or it can be different service or services. If they will have their own API's then they need to be accessed from outside.



For the given case I entered some tables and databases. Databases does not have any relations between them but they hold their reference_id of the table in other databases. They will communicate through their API's. In some cases all APIs use same database but in microservice manifest they say one service one database. so design it according to that. It is not whole diagram just a little part of it.

Event Campaign

event_campaign	
PK	**id**
	name
	lang_code
	shopping_center_id
	is_active
	start_date
	end_date

Shopping Centre

shopping_center	
PK	**id**
	name
	lang_code
	is_active

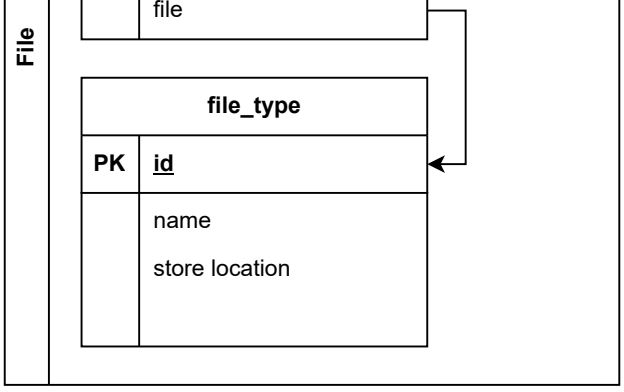
Report

reports	
PK	**id**
	name
	file_id
	file

Log

log	
PK	**id**
	name
	request
	response

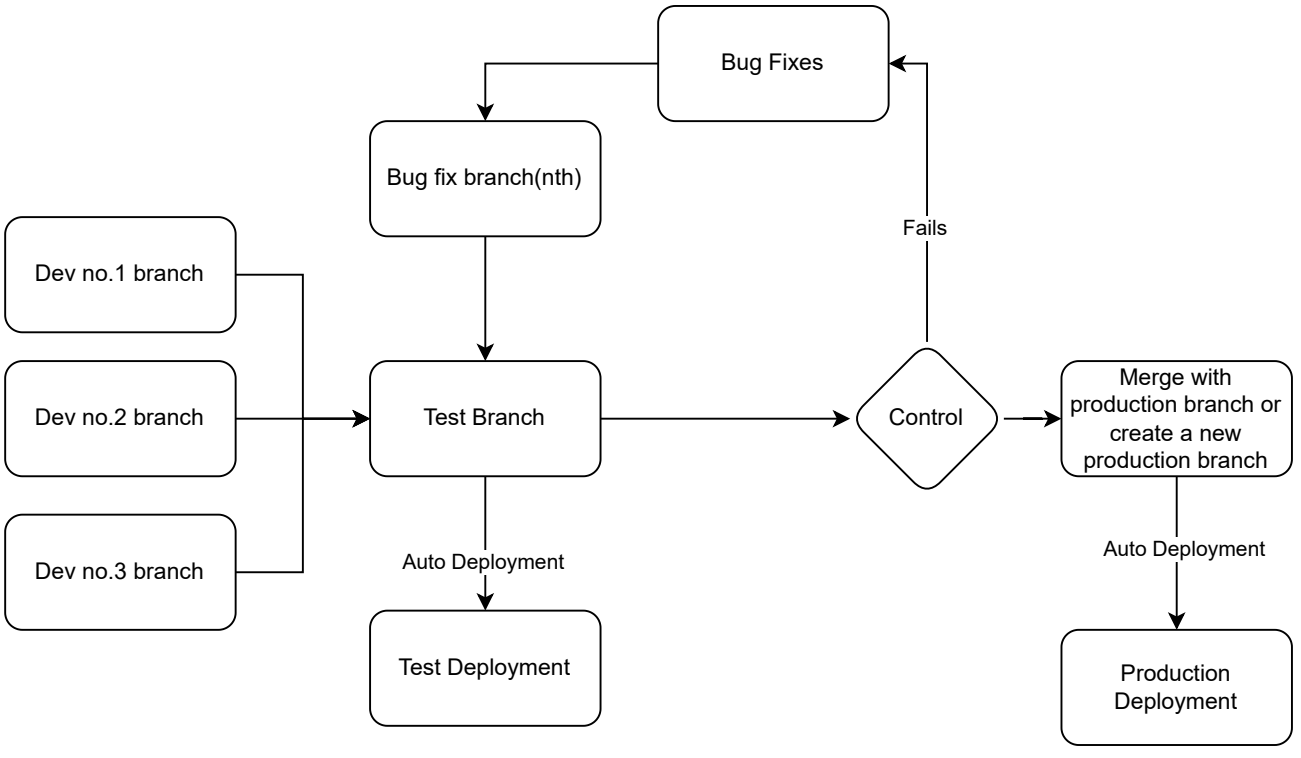
file	
PK	**id**
	name
	file_type_id



CI/CD

- Process will contain 3 stages. Development, Staging, Production. Commits will be done in specific branches of the requirement or story. Then when the development ends it will be merged into staging branch and deployed to test environment. When the tests are completed it can be merged into production env.
- Some source management tools allow for CI/CD so I will create a pipeline btw. production branch and env. to make automated deployments. If resources provided automated tests can be appliable.
-

CI/CD



Sprint Plan

- Backlog prioritization should be decided with product owner and team leads.
- Sprint plan can include new features for the product. It should be decided on which product sprint will aim to release.
- Every end of the sprint product should be ready to release.
- In my point of view one story and multiple subtasks is enough.
- For actions it will consist ToDo-In Progress-Test-ReadyToRelease steps can handle simple sprint process.
- A simple story should have design,frontend,backend subtasks. If provided in story accept criterias should be added while defineing story.
- Optimal sprint duration should be decided after 2 sprints. Usually perspective is one sprint is enough but 2 tryout sprints to be sure. To aim 2 week sprints and which team member handle how many story points or to see what is he or she good at can take up 2 sprint tryouts.
- When every sprint is completed team should gather around and anonymously enter what went well or what should be improved during the sprint. next sprint should start after that retro meeting.
- My approach to leading a team is be friend with everyone and inspect progress of the sprint daily basis. Team can ask any question to team leader. If team can become "team" it is always easy to manage and friendly env. keeps everyone's motivation high.

Security

- Security will be provided by Jwt token. Gateway will control token validation parameters and if the token is not authenticated request will never be forwarded to API. A middleware can sort this out. If wanted frontend can provide ip address and can be held in balck-list for further more security. If needed claims can be used for apı level authorization.
- In this design no one can send a request to backend api directly it will always go from gateway to API.
- In some cases if API needed to be reachable without gateway, token validation can be added to API separately to ensure security is provided for that API.
- On GDPR aspect. Application should get consent of the user to hold their personal information. Data should be removed if user want them to delete so pacifying an account is not an option if user want to his/hers information to be removed from system.