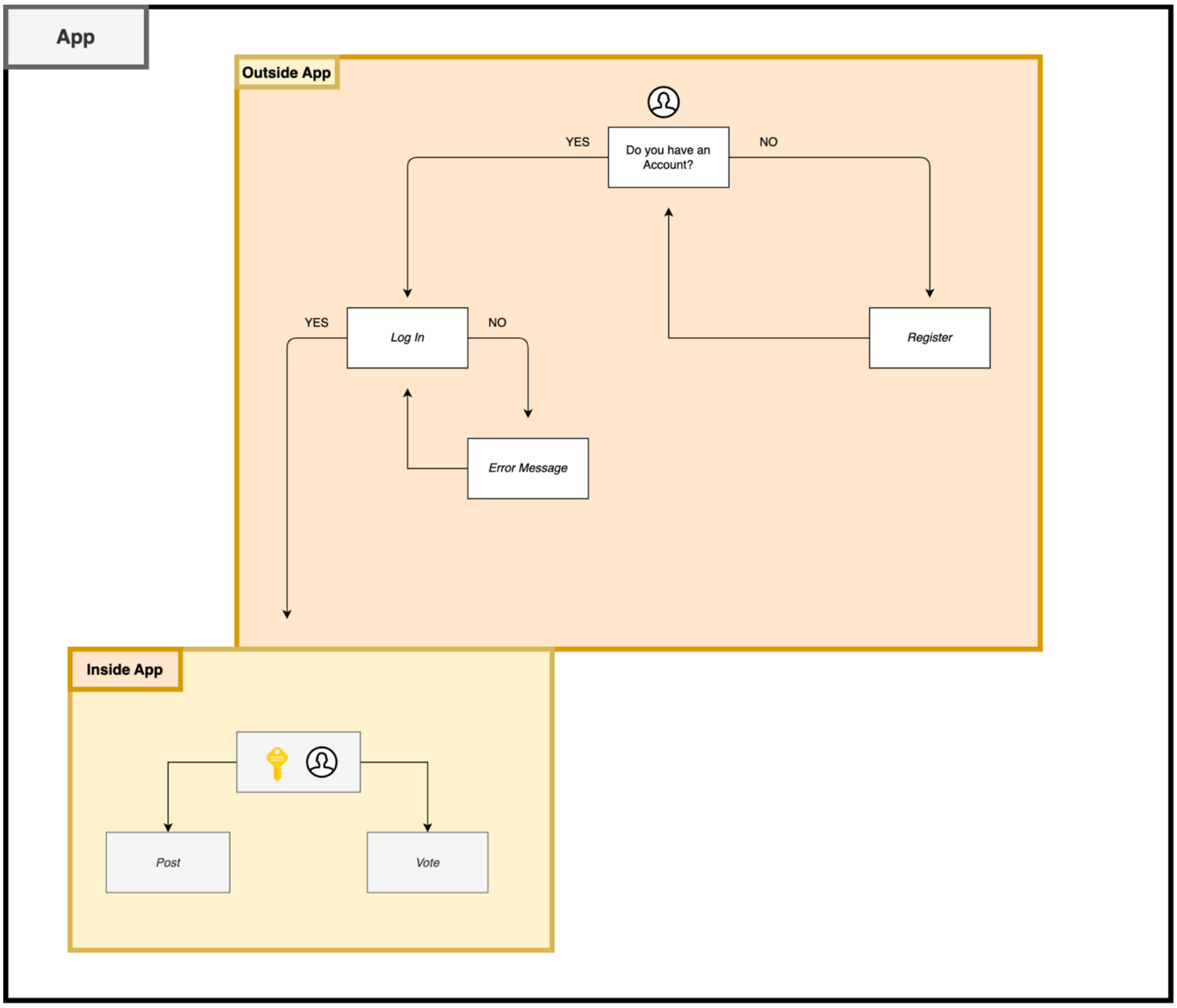
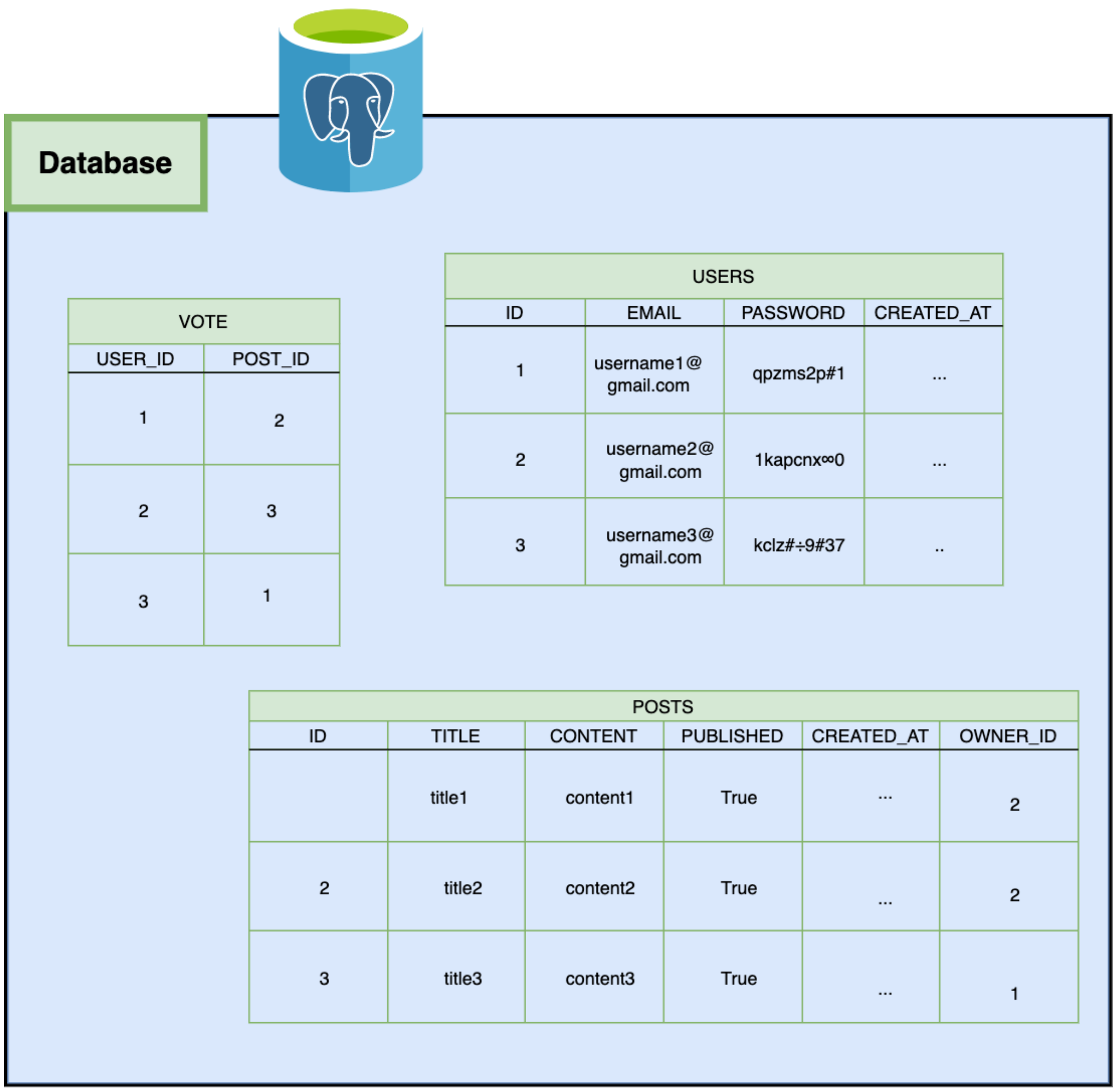
API Development with FastAPI

We will illustrate the general workflow of the API, in order to improve the clarity of the project for people reading it.

There are two main services, the Database and the App.





VOTE

USER_ID	POST_ID
1	2
2	3
3	1

USERS

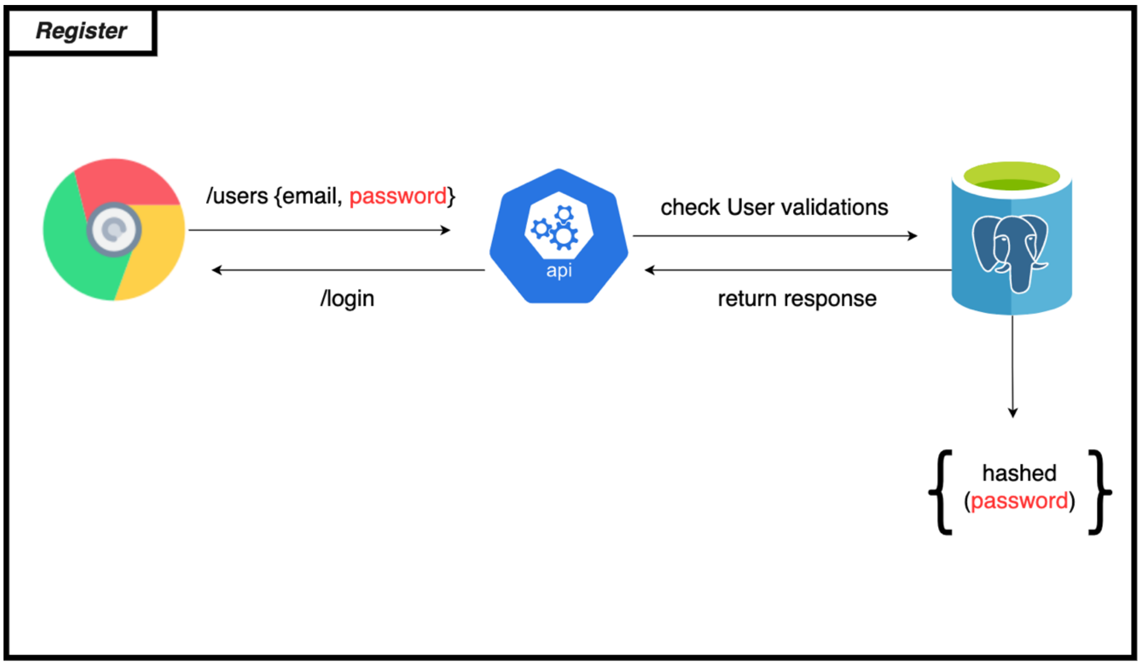
ID	EMAIL	PASSWORD	CREATED_AT
1	username1@gmail.com	qpzms2p#1	...
2	username2@gmail.com	1kapcnx∞0	...
3	username3@gmail.com	kclz#÷9#37	..

POSTS

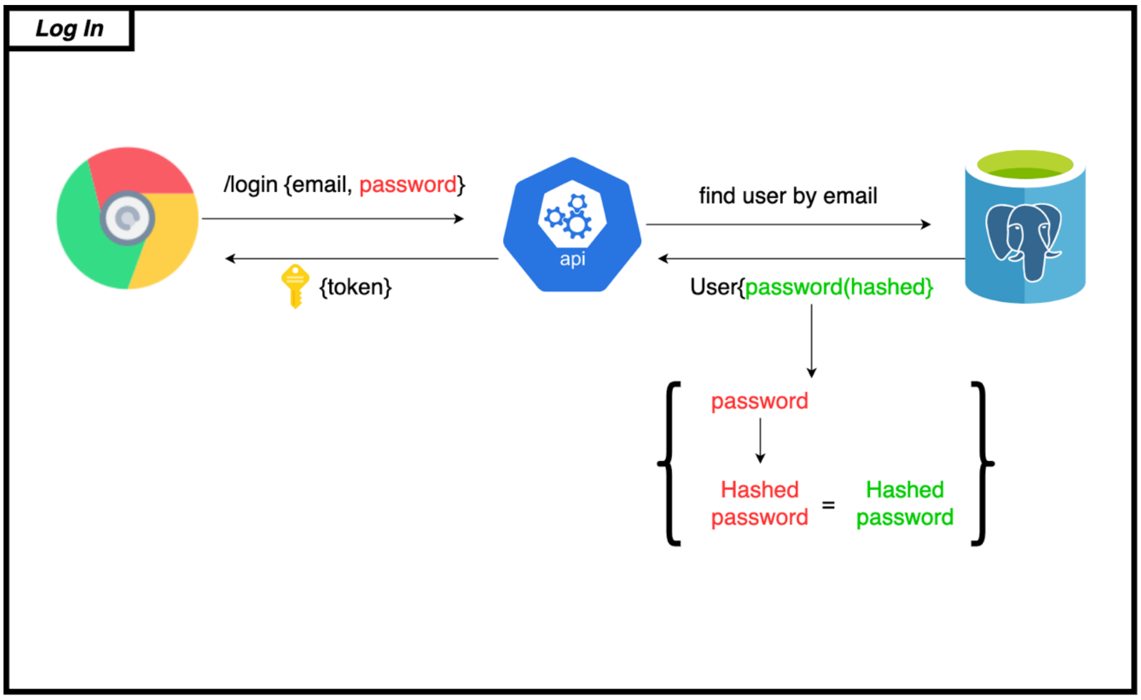
ID	TITLE	CONTENT	PUBLISHED	CREATED_AT	OWNER_ID
	title1	content1	True	...	2
2	title2	content2	True	...	2
3	title3	content3	True	...	1

As we can see on the App Diagram, we have several functionalities. More specifically, the User will be able to firstly Register and then Log In. Once logged in on the App, the Database will hold its User information and it will be able to do certain activities such as creating, updating or deleting a post and voting (like or dislike) them. Let´s start by briefly analyzing the User Registration, Log In and Authentication.

Just as an observation to avoid a possible confussion, on the App Diagram I separate the processes in two steps, “Outside” and “Inside” the App. This is not referencing the main FastAPI object commonly call as “App”, but what consumers tend to call as “App” which is the application itself.



Firstly, the User will have to create an account. To do so, it will have to pass some validations, such as no repetition on the email selected and so on. Once this criteria had been reached the person will have its account created and ready to be use whenever he wants. Our database will kept its email as username, the password (this is kept hashed because of security reasons) and finally a personal key identificatory or "id".



If you have already created your personal account you will be able to move on to this step. You load your personal account information, if its matches from one with our

database you will be able to browse around the app!. As I mentioned previously, the password are saved hashed, so we have to hash it first before doing the query.
Lastly, every User is Logged in has an Access Token (in this case I chose a JWT Token). This will enable it to perform requests such as the ones mentioned. Essentially, the objective of this are for security reasons, to double authenticate, this will last as much time I assign it to do so, in this case is 30 minutes. It's a good practice for any developer to implement this.