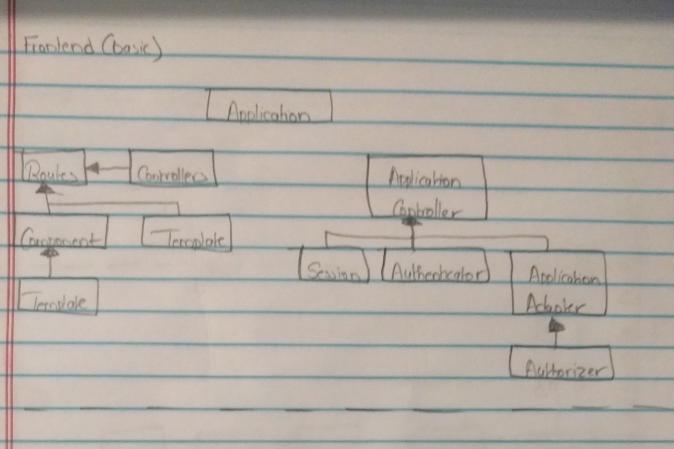
Backend Server «interface» Controller Token Controller User Controller Reservation Controller +generateToken(Request): json (+getRequest(): json +get(Request):json + Pailed Kesponse(): json /+ Create (Request): json +Creak(Requot): json + SuccessResponse(String): json + update (Request) = json + update (Request): json 1+ Validator (Request): json + Validator (Proquest): ison + Volidation (Request); ison + Send Frail (Requat) = Ison Profile Controller Facility Controller +get(Request) = json tgel(Request)= json tupdak(Request): joon +create(Request): json + validation (Request): jon 1 tupdale (Request): ison + validator (Request): ison Database ((interface)) Model Facilities Reservations Useo



The interface for the controller isn't acknowly an interface. It might be better labeled as a parent class. It includes basic functions for connection and doishers. It is really powerful since you can make a bunch of "sub" condrollers. that can easily update records in the DB tables, and pull date when needed

Each controller also includes validation of data so very request sent to the api automatically gets validated

As for the model interface, it includes some constants that define what

Can and cannot be done to the respective lable. An example of a constant is what dolo can be pulled and what

can be slored in the db

there is acheally a model interface on the Brontend that is very Similar. Fach model that exists on the backend, abo exists on the fronters The Frontend models are used to pape data from the backers