Guidelines:

1. The Angular Styleguide, here, will give you a head start: https://angular.io/guide/styleguide
2. Split your application into layers:

    [View Model (Component)] < --- > [Data Layer] < --- > [API Layer]

    1. **API Layer Structure**
        1. Should probably be in the Shared module, ex: shared/api as it can be used by anyone within the application;
        2. Should consist of DTOs (data transfer objects) and API Services.
        3. An API Service is an angular service that provides CRUD (Create, Read, Update, Delete) via http POST, GET, PATCH/PUT, and DELETE methods, against a restful API (aka 'resource')
        4. Each API service should have the following methods:
            1. findAll(): Observable<Response>
            2. find(): Observable<Response>
            3. create(): Observable<Response>
            4. delete(): Observable<Response>
            5. update(): Observable<Response>
    2. Models:
        1. Do not have to be in an Angular Module. However it's best if they are the same place you defined your Data Layer services.
        2. You should know the difference between a DTO and a view model. For each entity, example, customer, you would need to create interfaces as follows:
            1. CustomerDTO -> This interface should reflect what the API gives me when I ask for a list of customers.
               Example:
               interface CustomerDTO {
                 name: string;
                 surname: string;
                 id: string;
               }
            2. CreateCustomerDTO -> This interface should reflect what I can send to the API when I want to create a customer.
               It is very similar to the DTO in #1, however some fields may be different. For example, you do not send the ID when you create a new customer.
               Example:
               interface CreateCustomerDTO {
                 name: string;
                 surname: string;
               }

    3. UpdateCustomerDTO -> This interface should reflect what I can send to the API when I want to update a customer.
       Some business logic could prevent certain fields from appearing in an update altogether. Usually similar or identical to #2.
    4. AddEditCustomerViewModel -> This is the model that reflects what I'm doing inside my 'AddEditCustomerComponent', which can be a component that allows me to add or edit customers.
       The model should be placed in the same folder as the component.
       The component might want to add additional logic that is relevant only to the view.
       Example :
       ```
       interface AddEditCustomerViewModel {
         name: string;
         surname: string;
         id: string;
         isEditing: boolean;
       }
       ```
3. Data Layer:
    1. Data layer services consume the APIs from the API service, and convert these into DTOs.
    2. Possibly also error handling, but maybe this at a a later stage.
    3. Each Data layer service should have the following methods:
        1. findAll(): Observable<Array<DTO>>
        2. find(): Observable<DTO>
        3. create(): Observable<void>
        4. delete(): Observable<void>
        5. update(): Observable<void>
4. Components:
    1. Should be split across modules. The angular styleguide has a very good structure on how to split these up.