Architecture Planning Template

Dejan Guberinic

App Overview

App Features

Domain Security

Domain Rules

Logging

Services / Communications

Data Models

Feature Components

Shared Functionality

App Overview

- Overall Goals
  - Support viewing and editing customers
  - Support viewing orders
  - Secure customer edit form

- Key Requirements
  - Display customer with card/grid option
  - Support filtering, sorting and paging customers
  - Map customers (Google maps)
  - Customer editing support (CRUD)
  - Display orders with paging
  - Support login / logout with email / password

App Features

- Customers Feature
  - Display customers
  - Support card / grid modes
  - Dsiplay customers on map
  - Support filtering / paging / sorting
- Customer Feautre
  - Create / Update & Delete customer entity
  - Display customer details / orders / map
  - From provides validation
- Orders Feature
  - Display orders
  - Support paging
- Login / Logout feature

Domain Security

- Email / Password for initial release
- Consider tokens for future release?
  - HttpInterceptor could be used to set auth header
  - What option will be used for token issuer?

Domain Rules

- Each order must be associated with a customer
- Order totals should be shown for a given customer
- Customer edit form should validate data entry
- User must login to access customer edit form (route guard)
- Validate login credentials

Logging

- Create an Angular service for logging
- Consider using Azure AppInsights
  - Wrap existing AppInsights client SDK
  - Handle logging to different areas based on dev/stage/prod
    - Console logger
    - AppInsights logger

Services / Communication

- RESTful Service will be used (Java Spring, Node.js)
- Angular Services
  - Data service: Use HttpClient
    - Customers / Orders
    - Login / Logout (auth)
  - Sorting service (or pipe)
  - Filtering service (or pipe)
  - Logger service
  - Mediator / EventBus if needed
- HttpInterceptor will set token for HTTP requests if tokens used (GET/PUT/POST/DELETE/PATCH)

Data Models

- Application models / interfaces:
    - Customer model / interface
    - Auth model / interface
    - No order editing so include with Customer?

- Create a class or just use an interface on the client-side?

Customer Model

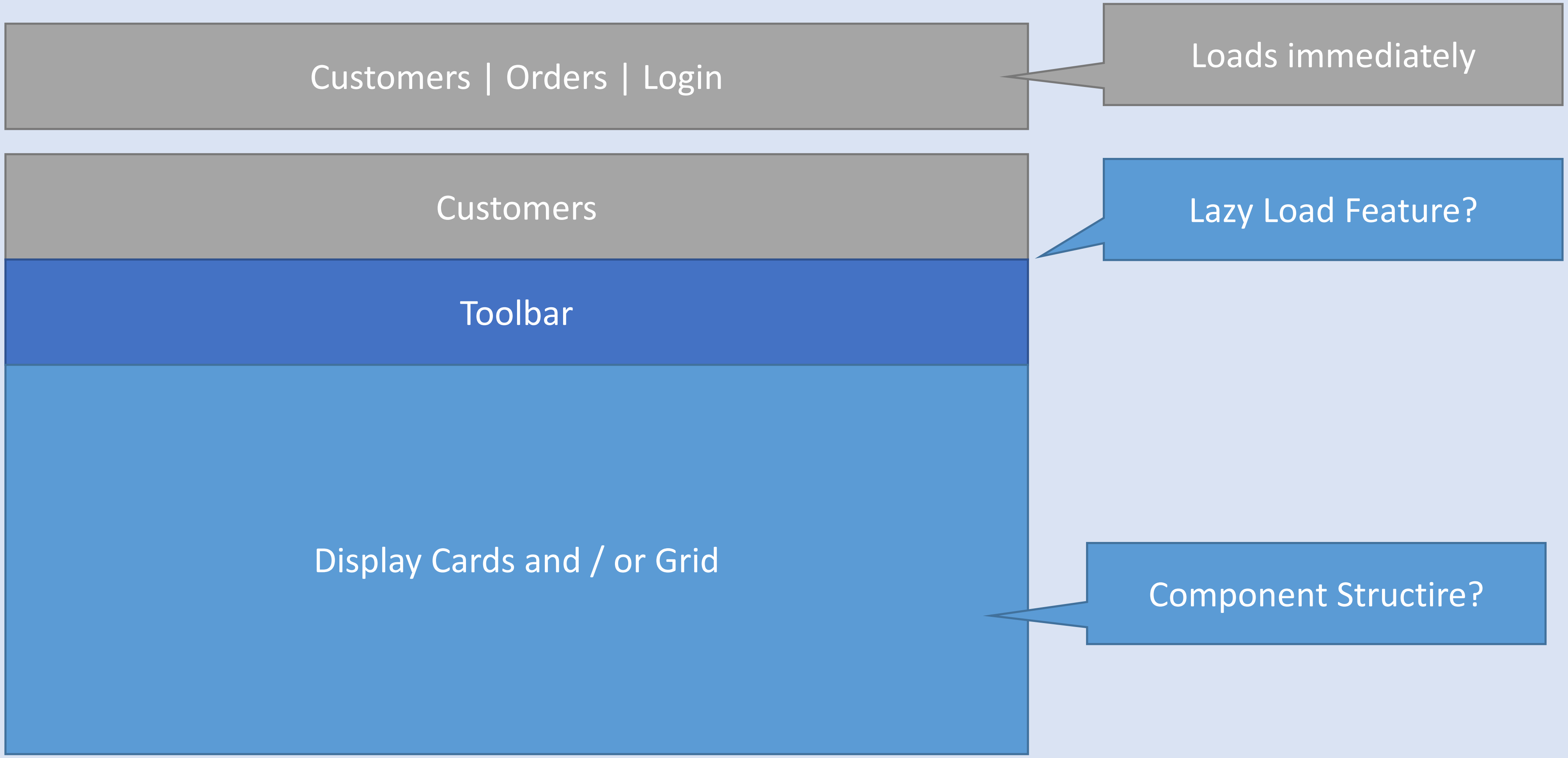
- Customer

Id	number
firstName	string
lastName	string
….	
Orders	Order[]

Feature Components

- Customers
  - Display / filter / sort / page customers (need data service)
- Customer
  - Create / read / update / delete customer (need data service and model needed for deletes)
- Orders
  - Display / page orders
- Login
  - Login form and logout functionality (need auth service)

Component UI Layout



Shared Functionality

- Toast / Growl (success / information)
- Modal dialog
- Google maps
- Pager
- Menu
- Grid / Card (used more then once?)

Shared Functionality (3rd party)

- Are we going to use any 3rd party components? Research a few options such as:
    - Prime Ng
    - Ng Bootstrap
    - Angular Material
    - Ag-grid