NTTData social networking site

This project has two separated data usages strategies and does not require real time data update and concurrency handling. Technologies like stateless web apps, workers services and denormalized data sets could be used for high availability.

**Project build with 2 stages:**

MVP - Proof of concept with frontend app and rest api service

V2 - High available and scalable social network

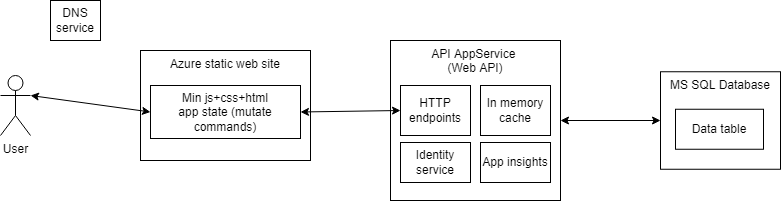
**Main components for each stage:**

1. MVP - CRUD (Slide 1 - MVP)

AppService for Frontend

AppService for Backend

MS SQL with Data table



2. V2 - Social network (Slide 2-SNS)

DNS

Azure Storage for static content

CDN

Load balancer

Identity service

AppService for Read

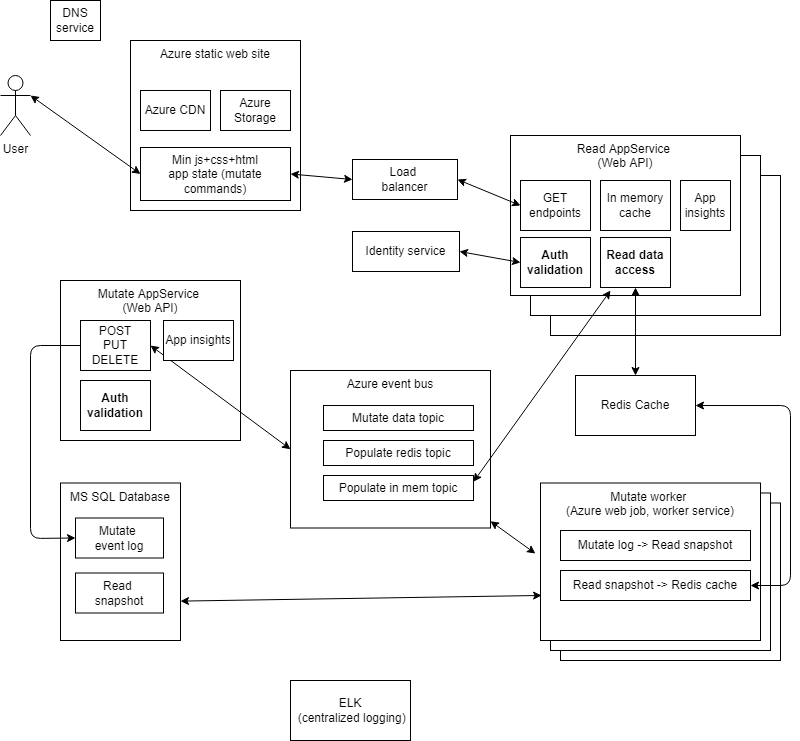
AppService for Mutate

AppService for Workers

MS SQL database for Data and event source

Azure service bus

Redis cache



Requirements

**Requirements to Frontend:**

- Produces minified js + css

- Has state management

- Change pages without reload

- UI validation

Probably good to use frameworks like ReactJs or Angular.

**Requirements to Backend:**

- REST support

- Server side validation

- Data access libraries for MS SQL server and Redis

- Event bus support (publish, subscribe)

- Horizontal scale

**Requirements to Workers:**

- Deploy to azure

- Logging

- Event bus support

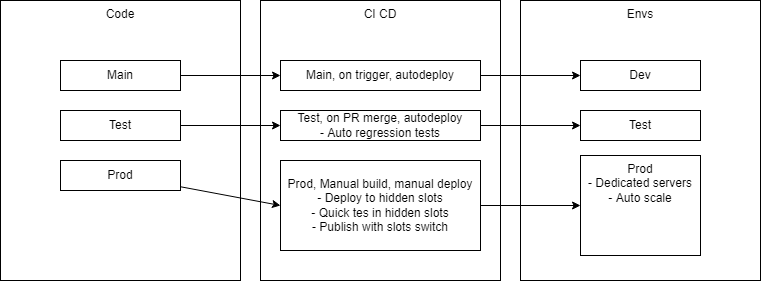
- Horizontal scale

Development

Branches: Main, Test, Prod and merges only via PRs. Gitflow started from Main.

Envs: Dev, Test, Prod

Pipelines: CICD with Azure DevOps



Main project structure - Mono repo. Here is proposition for files and folders structure:

*Assets*

*- Azure jsons, deployment configs*

*Frontend*

*- src*

*- dist*

*Reader API*

*- .NET Core App*

*Writer API*

*- .NET Core App with CQRS from start*

*- API*

*- Application*

*- Persistence*

*Workers*

*- Mutate worker*

*- Snapshot worker*

*Identity server*

*- .NET Core app with Identity Server 4*

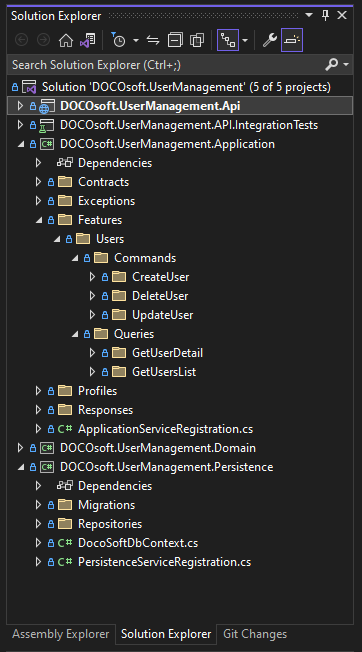
*Common*

*- Data contracts*

*- Data access interfaces*

*- Validation*

Web API application project structure



PS

Next steps:

1. Describe message contracts in Frontend and Backend projects and messages used on Service Bus
2. Describe testing strategy with Functional and Performance tests