TEKNISK PRØVE

Denne prøve har til formål at vise os hvordan du løser en given opgave og hvilke tanker du gør dig undervejs. Vi tager prøven meget alvorligt og den spiller en stor rolle i vores vurdering af dig som udvikler.

Det er ikke et krav at løsningen er klar til at blive lagt i drift, men det er vigtigt, at du har overvejet, og via koden viser, hvordan løsningen ville se ud hvis den blev gjort produktionsklar. Dette omfatter, men er ikke begrænset til, at du viser hvordan løsningen skal autotestes og hvilket kommentarniveau du synes er passende.

Du bestemmer selv hvilke teknologier og frameworks du vil bruge, men hvor relavant vil det være ønskeligt, hvis du udvikler løsningen i henhold til de retningslinier der er bveskrevet i ” Neas Architecture Commandments” (se side 2).

OPGAVEFORMULERING

Du har nogle sælgere som sælger varer til en række butikker.
Butikkerne er opdelt i distrikter der har navne, eks. Østjylland – Sønderjylland osv.
Hver sælger er knyttet til hver deres distrikt. For hvert distrikt er der én og kun én ansvarlig sælger (ALTID), og nogle gange er der også en eller flere sekundære sælgere.
I en overgangsperiode, hvis der mangler sælgere, kan en sælger godt have to distrikter som de er ansvarlige for.
Ikke alle sælgere har ansvaret for et distrikt, men et distrikt har altid 1 ansvarlig sælger.

OPGAVE 1

Lav tabeller der giver mulighed for ovenstående – så vidt muligt alle eller de fleste af forretningsreglerne skal afdækkes i constraints (relationer og null-værdier)

OPGAVE 2

Tilføj noget testdata til databasen (Distrikter og sælgere)

Lav en applikation (desktop/web) der viser alle distrikter i en liste. Når man klikker på et distrikt, ønskes en oversigt over sælgere knyttet til distriktet, samt de butikker som hører til distriktet.

OPGAVE 3

Lav mulighed for at tilknytte og fjerne sælgere fra distrikter. Det skal være muligt at angive om sælgere er primære eller sekundære.

Delopgaverne her består i:

Lav et datalag på server-siden, som kan hente data fra databasen (her skal du bruge ren SQL – ikke en ORM)

Lav en RESTful ASP.NET Core web service som bruger datalaget til at levere data til og fra databasen.

Lav en WPF eller AngularJs/Angular klient som kan hente data og vise det på en side.

NEAS ARCHITECTURE COMMANDMENTS

Design

- **Avoid monoliths**
  Small services / apps / systems
- **Single system data ownership**
  Only one system owns wind prognosis
- **DON'T use other systems databases**
  Use their WebServices

Communications

- **Systems expose/share data via Web Services**
  Data is available via HTTP GET
- **Systems receive data via Web Services**
  Data is sent via HTTP PUT
- **Systems send event notifications via**
  - **RabbitMQ** (*guaranteed delivery*)
  - **SignalR**

Technologies

- **Web:** Angular
- **Desktop:** C#, WPF
- **Web Services:** ASP.NET WebAPI + ASP.NET Core
- **Logging:** Serilog + ElasticSearch + Kibana
- **Deploy:** Jenkins + Octopus Deploy
- **Alerting:** ElastAlert, site24x7.com + PRTG
- **Database:** Microsoft SQL Server
- **IOC container:** Autofac, SimpleInjector
- **DB Access:** Dapper, Entity Framework
- **Shared DLLs:** Available from NuGet
- **All code (incl. stored procedures and views from the database) MUST be committed to a GIT repository**

Web Services

- **Protocol: REST via HTTP** (GET, POST and PUT)
- **Data format: JSON**
  Other formats are allowed, JSON is **required**
  Use Protobuf for high performance needs
- **URL must be api.neasenergy.com/...**
- **Must be available via the Data Portal**
- **Must be backwards compatible**
- **Automated End2End tests are required**
- **Use UTC for date/time values**
- **Use NEAS Market Model for 'key' values**
- **Windows authentication is required**
  Use the access management system for access outside Neas
  Use AD credentials for authentication inside Neas

Logging

- **Fatal** - Unrecoverable crash
- **Error** - Recoverable + Affects users
- **Warning** - Recoverable
- **Info** - Significant event
- **Debug** - Developer Information
- **Verbose** - Detailed debugging information

Critical Systems

- **Must have a failover strategy**
- **Must have continuous integration**
- **Must have a test system (w/o dependencies)**
- **Must have an external integration test system**
- **Must have a release procedure**