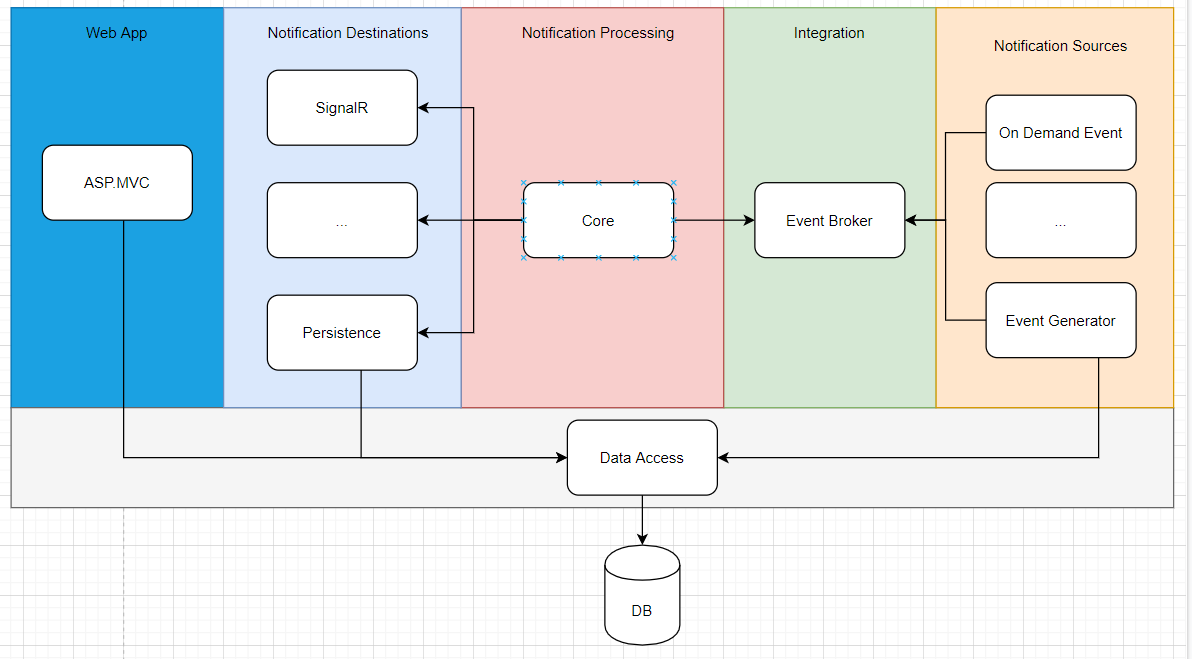
# Notification Center



We suppose the system is not microservice based, so the architecture is monolithically oriented.

## Technologies:

* ASP.MVC for the Web Application. The UI is relatively simple, so server-side rendering is appropriate.
* SignalR for supporting real time web notifications.
* IHostedService for the background job needed in Event Generator
* EF Core is used for data access
* MSSQL Server for database

## Components

* Notification sources – we might need to support multiple notification sources since information about the system is not presented
  + On Demand Event – used for on demand entity changes. Once the entity is being changed from other system module, an event will be sent.
  + Event Generator – background job used for generating custom data related event. It will crawl the database every hour, will check if there are expired certificates and will produces events.
* Integration – the integration component is used for decoupling the different system module/components with the help of message communication
  + Event Broker – supporting the message transferring between the components.
* Notification processing – the component will receive, process and route all published messages by the sources
* Notification destinations – the component is responsible for the correct message delivery
  + SignalR – used for real time notifications
  + Persistence - used for saving the messages in the storage
* ASP.MVC – web-based application used for visualizing the needed information
* Data Access – standard data access layer used for decoupling the business from the data logic

The event message communication introduces big chunk of flexibility and makes the solution extendable during its evolution. Therefore, adding new notification sources and destinations becomes an easy task.

# Database diagram

