• Business Central system topology in Saas.

* Application Tier - The application tier hosts the Business Central application
* Web Server Tier - The web server tier in the SaaS topology consists of multiple web servers hosted in Azure.
* Database Tier - The database tier stores the Business Central application data. In SaaS, Microsoft manages the database infrastructure, ensuring data security, availability, and backups
* Azure Datacenters - Business Central SaaS is hosted there

Also, other Azure services are accessible.

ALSO, Microsoft handles all updates (security, system, etc.) for BC SaaS. So, the only thing you have to be responsible for is good internet connection (which sounds as a joke, if someone remembers last winter blackouts)

• What is the life cycle of a Business Central extension in Saas?

1. Of course, the first stage of extension`s life cycle is it`s development (done from visual studio code as a development environment, published from there. Also, other tools are used)
2. Then goes deployment. BC`s cloud solution updates by tenant from time to time, so extension better to be published from extension management page (and also installed from there) to better not to be dropped with the next update.
3. Updates. Sometimes BC`s user may ask for additional customizations for an extension.
4. Retirement. Client wants to move to another system or to use new extension instead of the old one.

• What are Business Central’s integration patterns (names, usages)? What is the main difference

between them?

Integration patterns are different approaches of connecting Business Central with other systems to pass some certain data or add another segment of functionality. There are several integration patterns:

* Migration Pattern – Moving data from one system to another (at a specified moment of time or under specified conditions).
* Broadcast Pattern – Moving data from one to several systems simultaneously in real time.
* Aggregation Pattern – Collecting data from several systems and copy it to one Business Central system.
* Bi-directional Synchronization – Making two datasets from two systems work as one (synchronize them).
* Correlation Pattern – Bi-directional Synchronization with prior identification of intersections of two datasets (with the following processing only in this scope)

• What is Branching strategy? (Usages examples are needed)

Basically, this strategy involves git technology as it`s main tool.

For each purpose (for each dev, usually) on project there is created a branch with a copy of an extension which is being developed by a team. Some individual (related to the project) task is being done in every of branches. When the task for a certain branch is being finished the pull request is being created to pull all changes from this branch to the main one. Main dev revises it and decides whether changes are good to go to the main branch or something has to be done to them (and declines the pull request in this case). Also, changes from the main branch has to be pulled to secondary branches from time to time so devs would work with the same stage of extension`s development.

As a usage example any project which involves a team of developers (more then one) will do.

Additional info on tasks:

modified fields on customer Card:

* Before:

A screenshot of a computer screen

Description automatically generated

* After:

A screenshot of a computer

Description automatically generated

Modified Standard Report – “Customer - Order Summary”(107):

* Before

A screenshot of a report

Description automatically generated

* After