# Глава 3. Изграждане на облачната система за производствено предприятие HeidelbergCement AG

## 3.1. Обща характеристика на дейността на компанията HeidelbergCement

HeidelbergCement е немска мултинационална компания за строителни материали със седалище в Хайделберг, Германия. След придобиване на 45% акционерно участие в Italcementi, HeidelbergCement става производител номер едно на строителни инертни материали, номер 2 в цимент и номер 3 бетон в световен мащаб. В Forbes Global 2000 за 2020 г. HeidelbergCement бе класирана като 678-ата най-голяма публична компания в света. Дейността на фирмата е в около 60 страни с 57 000 служители, работещи в 3 000 производствени обекта. HeidelbergCement управлява 139 циментови завода с годишен циментов капацитет от 176 милиона тона, повече от 1500 места за производство на готов бетон и над 600 кариери за инертни материали.

### 3.1.1. Основни бизнес процеси в компанията

Продуктите на компанията се използват за изграждане на къщи, инфраструктура, търговски и промишлени съоръжения, като по този начин отговарят на нуждите на нарастващото световно население за жилища, мобилност и икономическо развитие. Основната дейност включва производство и дистрибуция на цимент, инертни материали, готови бетонови смеси и асфалт.

### 3.1.2. Промяна на индустрията към дигитализация

Дигитализацията е един от стълбовете на трансформацията. Тя преминава през всички бизнес линии и операции. Дигитални продукти, насочени към клиентите, целят да помогнат за успех в основния бизнес. (von Achten, 2022)

HeidelbergCement се фокусира върху промяната на индустрията на тежките строителни материали към дигитализация. Това ще подпринесе за безопасност и устойчивост към работниците.

## 3.2. Избор на технологични средства за разработка и операции

Тази подточка, ще се опише едно от най-важните решения, най-вече защото е почти необратимо. Основни съображения за изпълнение на задачата:

* + Общност – Stack Overflow;
  + Популярност – Google Тенденции;
  + Краен срок – напредналите технологии отнемат повече време;
  + Поддръжка – развивитие с нови инструменти;
  + Продукти – използване на съществуващи инструменти;

<https://www.fgdc.gov/initiatives/50states/newspbp/EconomicJustification_ROI-CBA-Tutorial_v2_052809_FinalVersa.pdf>

Сравенение на Back-end и сервизни технологии



Сравнение на Mobile technologies



TODO: database technologies

## 3.3. Приложение на избраните технологии за изграждане на инфраструктурата в облачно базирана среда

По примери и указания от глава 2, тази подточка разглежда осъществяването на опростен във функционално отношение, облачен продукт, демонстриращ използването на .NET, Docker, Kubernetes в облачната среда на Microsoft Azure.

### 3.3.1. Монолитен подход за изграждане

Повечето традиционни .NET приложения се внедряват като единици, съответстващи на изпълними файлове или казано по друг начин уеб приложения, работещи в рамките на един домейн на IIS сървър. Този подход е най-простият модел за внедряване и обслужва добре много вътрешни и по-малки публични приложения. Това са така наречените монолитни приложения - напълно самостоятелни по отношение на своето поведение. Могат да взаимодействат с други услуги или хранилища на данни в хода на извършване на своите операции, но ядрото на тяхното поведение се изпълнява в рамките на собствен процес и обикновено цялото приложение се разгръща като самостоятелна единица. Ако такова приложение трябва да се мащабира хоризонтално, обикновено то се дублира върху множество сървъри или виртуални машини. Това са приложения от тип „всичко в едно“. В тази архитектура, цялата логика на приложението се съдържа в един проект, компилиран и внедрен като самостоятелна единица. Шаблонът на нов ASP.NET Core проект, независимо дали е създаден във Visual Studio или от командния ред, започва като прост монолит „всичко в едно“. Той съдържа цялото поведение на приложението, включително логика за визуализация, бизнес и достъп до данни. Разделянето на логиката се постига чрез използването на папки.

На фиг. 2 е показан примерен дизайн на монолитно приложение. Модулите, отговарящи за първоначалните изисквания на приложението са разгледани в 2ра глава.

Diagram

Description automatically generated

***Фиг. 2. Традиционен монолитен дизайн***

Бизнес логиката е разпръсната между моделите и класовете на услуги без ясна индикация. Тази липса на организация на ниво проект често води до т.нар. „спагети код“. За да се справят с тези проблеми, приложенията често се развиват в много-проектни решения, където всеки проект отговаря на определен слой на приложението. Чрез организиране на кода в слоеве, общата функционалност на ниско ниво може да бъде преизползвана. Тази повторна употреба е от полза, защото показва, че трябва да се пише по-малко код и стандартизира една реализация.

Макар и просто, монолитното решение за един проект има някои недостатъци: когато размер и сложността на проекта нарастват, броят на файловете и папките също ще продължи да расте. Много успешни приложения, които съществуват днес, са създадени като монолити. С течение на времето, обаче, се наблюдават някои слаби точки като:

• Новите промени могат да имат нежелани и „скъпи“ странични ефекти;

• Новите функции стават трудни, отнемащи време за прилагане;

• Всяка версия изисква пълно обновяване на цялото приложение;

• Един нестабилен компонент може да срине цялата система;

### 3.2.2. Подход за изграждане, ориентиран към микросървисн

За да реши горе описаните, но и много други, проблеми, следва да разгледаме ориентирания към услуги, архитектурен стил. Това е подход за изграждане на сървърно приложение като набор от малки, но високо-качествени подуслуги. Съотвено, клиентите, на сървърните услуги, могат да бъдат отделни приложения, които да се поддържат и управляват самостоятелно. Всяка услуга работи в собствен процес и комуникира с други процеси, използвайки различен тип и вид протоколи, разгледани в подглава 2.4. Всеки микросервис притежава специфична бизнес способност, трябва да бъде разработван автономно и да може да се разгръща независимо. Предимства на това архитектурно решение са:

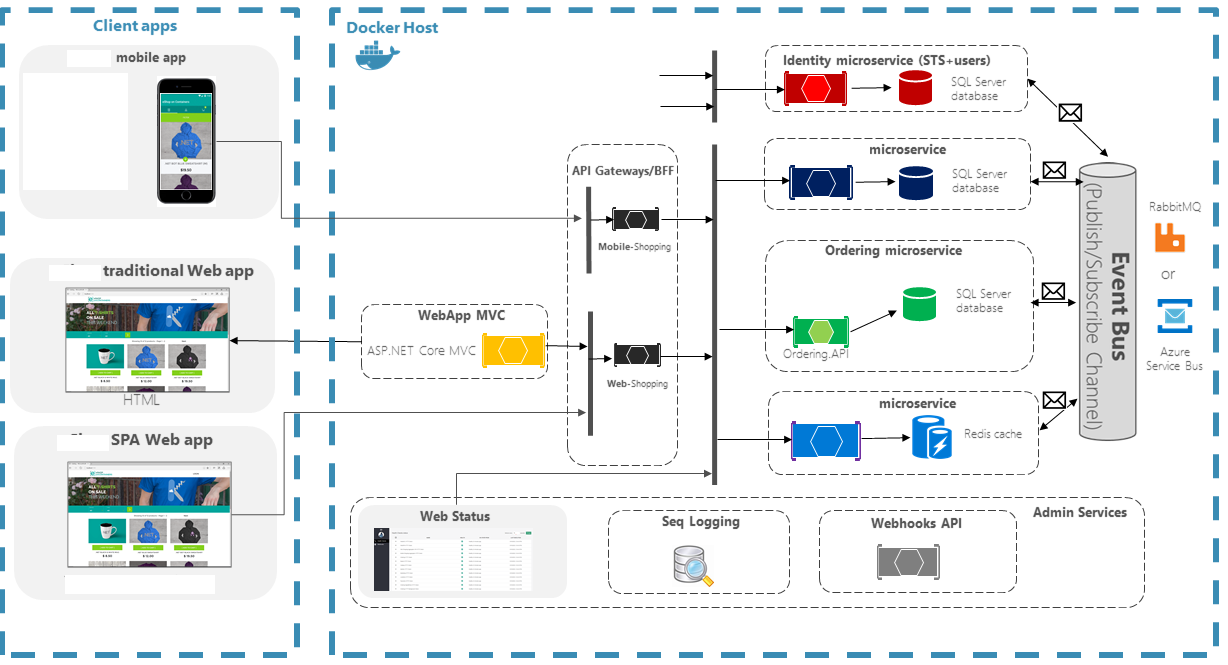
• Всяка микроуслуга може да бъде проектирана, разработена и внедрена независимо една от друга, което осигурява възможно за независима работа по отделни области на приложението.

• Работата може да бъде дистрибутирана между отделни екипи.

• Проблемите са по-изолирани.

• Позволява използването на различни технологии.

На фиг. 2 е показан примерен дизайн на микросървизно приложение. Всяка услуга, отговаря на изисквания разгледани в 2ра глава.



***Фиг. 2.2.*** *Диаграма от високо ниво на системата.*

### 3.2.3. Софтуерното внедряване и поддръжка в облачна среда

За изграждане, доставка и изпълнение на системи, изградени както като монолитни приложения, така и като ориентирани към услуги, се препоръчва използването на контейнеризирани технологии. Контейнеризацията е подход, в сферата на разработката на софтуер, при който кодът на приложение, всички негови зависимости и конфигурации са пакетирани в двоичен файл, наречен изображение. Изображенията са „шаблони“ само за четене и се съхраняват в регистър, който работи като хранилище или библиотека за изображения. Изображението се трансформира в работещ екземпляр на контейнер, който може да се стартира, спира, премества и изтрива. Създават се контейнери за различните части от приложението: уеб услуга, база данни, кеширане и др. Точно както транспортните контейнери позволяват транспортирането на стоки, независимо от товарите вътре, софтуерните контейнери се възприемат като стандартна единица за внедряване на софтуер, която може да съдържа различен код и зависимости. Контейнеризирането на софтуера дава възможност на разработчиците и ИТ специалистите автоматично да подновяват новите промени в различни среди. Контейнерите също така изолират приложенията едно от друго в споделена операционна система. Приложения се изпълняват върху хостът на контейнерите. От гледна точка на приложението, инстанцирането на изображение означава създаването на контейнер. Друго предимство на контейнеризацията е мащабируемостта. Разширяването става бързо: създават се нови контейнери за краткосрочни задачи. Контейнерите предлагат предимствата на изолация, преносимост, гъвкавост и контрол в целия жизнения цикъл на приложението. Azure предоставя услуги, които могат да помогнат за постигане на много неща, варирайки от обикновени, като създаване на ново приложение с база от данни – до по-развити  като създаване на работни потоци за непрекъсната интеграция (CI) и внедряване (CD). Това са само няколко примера за някои често срещани работни похвати. Много от тях трябва да бъдат създадени индивидуално, но облачната инфраструктура предлага  всичко това като услуги. Силата на облака е, че ресурсите са невероятно устойчиви, малко вероятно е аварийно да спрат работа, тъй като центровете за данни са разположени по целия свят, състоящи се от десетки хиляди сървъри. Ако един сървър се повреди, друг поема управлението. Един от най-убедителните аргументи в полза на облака е, че може да разширява мащаба на услуги и ресурси почти безкрайно, в определени моменти, като например "Черен Петък" или голяма маркетингова кампания с промоции и намаления на артикули. Също така, когато натоварването намалее, мащабът може да се намали до обикновените си параметри. Уважавани и опитни облачни доставчици като Microsoft разпознават моделите на използване на нормалните потребители и тези на злонамерените. Инфраструктурата е предпазена от най-често срещаните атаки. Интелигентни инструменти за наблюдение, алгоритми за обучение и изкуственият интелект предоставят възможност да откриват атаки. При стартиране на приложения в Azure eдно от първите решения, които трябва бъдат вземети, са планираните за използване услуги:

• Azure App Services - eдин от най-лесните и мощни начини за хостване на приложения. Той е предпочитан при монолитната архитектура. Услугите са достъпни и работят в 99,95% от времето. Споделят мощни функции като автоматично мащабиране, внедряване с нулев застой и лесно удостоверяване, позволяват отстраняването на грешки в приложението докато работи в производствена среда (със Snapshot Debugger). По подразбиране приложението ще бъде достъпно в интернет, без да е необходимо да се настройва име на домейн или да се конфигурира DNS. Работи много добре с контейнери.

• Azure Virtual Machines - позволява преместване на съществуващи приложения от виртуални машини, които вече се изпълняват във център за данни. Има много предварително дефинирани изображения, които могат да бъдат използвани като Windows Server, който работи с IIS и има инсталиран и предварително конфигуриран ASP.NET на него, както и собствени софтуерни лицензи (като за SQL Server). Услугата е подходяща за мигриране на т.нар. „наследена система“, която да бъде използва като подсистема или източник на данни.

Следната таблица представя услугите и техните най-чести случаи на употреба:

Chart, scatter chart

Description automatically generated

***Фиг. 8. Представя кои услуги на Azure са подходящи за различните типове.***

# Използвана литература

1. <https://www.forbes.com/companies/heidelbergcement/>
2. <https://www.forbes.com/lists/global2000/?sh=1939f7c65ac0>