Презентација 5:

1. Што е Data Strategy (Стратегија за податоци):

- Data Leadership значи разбирање на односот на организацијата со податоците и користење на сите достапни алатки за исполнување на бизнис целите.
- Клучни прашања: Кои се целите на организацијата? Како податоците можат да ги поддржат тие цели? Како се собираат, користат и подобруваат податоците?1

2. Data Architecture (Архитектура на податоци):

- Почнувајте од највредните податоци и како тие придонесуваат кон главните цели на организацијата.
- Архитектурата треба да биде флексибилна и да овозможи раст и промени во организацијата.
- Податоците треба да бидат достапни во реално време за клучните одлучувачи.
- Визуелизацијата и претставувањето на податоците се важни за нивна употреба.1

3. Клучни улоги во Data Architecture:

- Data Architect: дефинира визија и стандарди за податоци.
- Project Manager: управува со проекти за промена или создавање на нови податочни текови.
- Solution Architect, Cloud Architect, DBA/Data Engineer, Data Analyst, Data Scientist: секој има специфична улога во дизајнот, имплементацијата и користењето на архитектурата на податоци.1

4. Главни рамки (frameworks) за архитектура на податоци:

- DAMA-DMBOK 2: стандарди и најдобри практики за менаџмент на податоци.
- Zachman Framework: дефинира слоеви и аспекти на архитектурата преку прашањата Who, What, When, Where, Why, How.
- TOGAF: методологија за развој на корпоративна архитектура, со посебна фаза за Data Architecture.1

5. Клучни компоненти на Data Architecture според TOGAF:

- Дефинирање на главни типови и извори на податоци.
- Разбирање како податоците се користат во процесите и услугите.

- Управување со миграција, интеграција и трансформација на податоци.
- Воведување на стандарди и системи за управување и контрола на податоците (Data Governance).1

6. DAMA-DMBOK 2 – 11 области на менаџмент на податоци:

- Data Governance, Data Architecture, Data Modeling & Design, Data Storage & Operations, Data Security, Data Integration & Interoperability, Documents & Content, Reference & Master Data, Data Warehousing & Business Intelligence, Metadata, Data Quality.1

7. Методологии за интеграција на податоци:

- Manual Integration: рачна интеграција, погодна за мали и едноставни проекти.
- Middleware Integration: софтвер за комуникација меѓу стари и нови системи.
- Application-based Integration: апликации кои автоматски ги интегрираат податоците.
- Uniform Access Integration: овозможува унифициран пристап до податоците без нивно копирање.
- Common Storage Integration: создавање и складирање на копија од податоците за напредна анализа.1

8. Кога да се користи која интеграција:

- За едноставни анализи и мал број извори – Manual.
- За автоматизација меѓу стари и нови системи – Middleware.
- За комплексна анализа и повеќе системи – Application-based, Uniform Access или Common Storage.1

9. Алати за интеграција на податоци:

- Примери: Hevo Data, IRI Voracity, Xplenty, Informatica, Microsoft Azure, Talend, Oracle, IBM.
- Секоја алатка има различни карактеристики, конектори и можности за автоматизација и анализа.1

10. Зошто е важен стандардизиран пристап?

- Подобра ефикасност, пониски трошоци, усогласеност со регулативи, олеснување на раст и промени, и конзистентност во менаџментот на податоци.

Презентација 6:

1. Основи на системи за управување со бази на податоци (DBMS)

- DBMS се користат за ефикасно чување и пристап до податоци, најчесто за OLTP (On-Line Transaction Processing), што значи дека се оптимизирани за брзи трансакции, внесување, ажурирање и бришење на податоци.
- OLTP системите содржат тековни, „активни“ податоци, додека старите податоци се архивираат.
- Обезбедуваат безбедност и интегритет на податоците, како и можности за извештаи преку јазик за пребарување (на пример, SQL)1.

---

2. Релациски модел

- Податоците се нормализирани во повеќе табели за да се минимизираат дупликациите и да се олесни управувањето.
- Секоја табела се однесува на одреден ентитет (на пример, студент, класа).
- Примарниот клуч е уникатен идентификатор за секој запис во табелата1.

---

3. Проблеми со хетерогени извори на информации

- Различни интерфејси, различни формати на податоци, дупликати и неконзистентни информации се чести проблеми при интеграција на податоци од повеќе извори.
- Големите организации често страдаат од „вертикална фрагментација“ (stove pipes), каде што системите се развиени според потребите на одделни апликации1.

---

4. Што е Data Warehouse?

- Data Warehouse (DW) е централизирано складиште на интегрирани, историски податоци, добиени од различни извори, оптимизирани за анализа и извештаи, а не за секојдневни трансакции.
- Главни карактеристики:
    - *Subject-oriented* (организирани по субјекти, како продажба, производи)

- *Integrated* (интегрира податоци од повеќе извори)
- *Time-variant* (содржи историски податоци)
- *Nonvolatile* (податоците ретко се ажурираат директно)1.

---

5. Data Warehouse vs. OLTP

Карактеристика	**Data Warehouse (OLAP)**	**OLTP (DBMS)**
Обем на податоци	Голем (Gb, Tb)	Мал до голем
Тип на податоци	Историски, агрегирани	Тековни, детални
Операции	Читање, анализа	Внес, ажурирање, бришење
Фреквенција на ажурирање	Ретко, по распоред	Во реално време
Организација	По субјект	По апликација
Корисници	Аналитичари, менаџери	Оперативен персонал

---

6. Архитектури на Data Warehouse

- *Single-layer*: Сите податоци се чуваат еднаш (виртуелен warehouse).
- *Two-layer*: Комбинира реално време и изведени податоци (најчест во индустријата).
- *Three-layer*: Вклучува трансформација на податоци во повеќе чекори1.

---

7. Мултидимензионални податоци и Dimensional Modeling

- *Measures*: Нумерички податоци (продажба, профит).
- *Dimensions*: Бизнис параметри (време, регион, производ).
- *Dimensional modeling* користи *star schema* (факт табела во центар, опкружена со димензионални табели).

- Факт табелите содржат мерки, димензионалните табели содржат описни информации1.

---

8. OLAP (On-Line Analytical Processing)

- OLAP е „front-end“ за data warehouse, овозможува лесна анализа на податоците преку мултидимензионални операции: roll-up, drill-down, slice, dice, pivot.
- OLAP користи „data cube“ за брзи и флексибилни анализи1.

---

9. Проблеми со интеграција и интегритет на податоците

- Различни имиња за исти податоци, различни податоци со исти имиња, дупликати, различни клучеви, празни или невалидни полиња се чести проблеми при интеграција1.

---

10. Модерни облачни решенија за Data Warehousing

- *Redshift* (Amazon): Shared-nothing архитектура, високи перформанси, но compute и storage не се скалабилни независно.
- *Snowflake*: Shared-storage, compute и storage се скалабилни независно, поддржува директно прашање на неструктурирани податоци (JSON).
- *Azure SQL Data Warehouse*: Shared-storage, еластични ресурси, поддржува Polybase за big data.
- *Google BigQuery*: Одвоени storage и compute, автоматизирана скалабилност1.

---

11. Таблично партиционирање и паралелизам

- Партиционирање на табели го распределува товарот и овозможува паралелна обработка на податоци (hash, round robin, range)1.

---

12. Практична примена

- Data warehouse се користи за:

- Информациска обработка (извештаи, статистика)
- Аналитичка обработка (OLAP)
- Data mining (откривање скриени шеми и трендови)

Презентација 7:

1. Што е Cloud Computing?

- Cloud computing е модел за испорака на компјутерски ресурси (сервери, складирање, апликации, backup, безбедност) преку интернет, како услуга на барање. Овозможува флексибилност, скалабилност, достапност и заштеда на ресурси1.
- Основни карактеристики се: ресурси се достапни од било каде, плаќање според користење, автоматско скалирање, и услуги се хостирани на далечна инфраструктура1.

---

2. Архитектура и карактеристики на Cloud Computing

- Масовна скалабилност, географска дистрибуција, виртуализација, ориентација кон услуги, ниска цена на софтвер, напредна безбедност1.
- Клучни карактеристики: On Demand Self-Service, Broad Network Access, Rapid Elasticity, Resource Pooling, Measured Service1.

---

3. Видови Cloud услуги (Cloud Flavors)

Вид	Опис
SaaS	Software as a Service – софтвер се користи преку интернет (пример: Google Docs, Salesforce)1.
PaaS	Platform as a Service – платформа за развој и хостирање апликации (пример: Azure, Salesforce Platform)1.
IaaS	Infrastructure as a Service – инфраструктура (сервери, складирање) како услуга (пример: Amazon AWS)1.

Вид	Опис
DaaS	Desktop as a Service – цел десктоп се користи како услуга преку облак1.

---

4. Модели на cloud deployment

Модел	Опис
Public Cloud	Ресурси достапни за јавноста преку интернет, управувани од трета страна1.
Private Cloud	Ресурси се користат само од една организација, обично on-premise1.
Hybrid Cloud	Комбинација од public и private cloud, за баланс на контрола и скалабилност1.
Community Cloud	Заедничка инфраструктура за повеќе организации со слични потреби1.

---

5. Предности на Cloud Computing

- Намалени трошоци за хардвер и софтвер, бидејќи се користат web-based апликации1.
- Подобри перформанси, инстантни ажурирања, неограничен капацитет за складирање, поголема доверливост на податоци (backup во облак)1.
- Универзален пристап до документи, полесна колаборација, независност од уред, последна верзија на документи секогаш достапна1.

---

6. Недостатоци и предизвици на Cloud Computing

- Потребна е постојана интернет конекција; без неа нема пристап до услуги и податоци1.
- Ограничени функционалности во споредба со традиционални desktop апликации, потенцијално побавно работење1.
- Безбедност и приватност: ризик од неовластен пристап, губење на податоци, зависност од трета страна1.

- Проблеми со компатибилност, различни протоколи и API-и кај различни cloud провајдери1.

---

7. Виртуализација

- Виртуализацијата овозможува повеќе оперативни системи (виртуелни машини) да работат на еден физички сервер, споделувајќи ги ресурсите1.
- Клучни поими:
    - Host Machine – физичкиот сервер
    - Hypervisor – софтвер кој управува со виртуелните машини (VMs)
    - Guest OS – оперативен систем кој работи во VM1.
- Придобивки: намалени трошоци, изолација, лесна миграција, независност од хардвер, брзо креирање и бекап на машини1.

---

8. Врска меѓу Cloud Computing и Виртуализација

- Cloud computing го користи концептот на виртуализација за да понуди ресурси како услуга: не мора да поседуваш хардвер, туку изнајмуваш виртуелни сервери според потреба1.
- Ова овозможува скалабилност, флексибилност и оптимизација на трошоци1.

---

9. Водечки cloud провајдери (2020)

Провајдер	**Пазарен удел**
Amazon AWS	32%
Microsoft Azure	18%
Google Cloud	8%
IBM Cloud	5%

Провајдер	Пазарен удел
Alibaba Cloud	5%

---

10. Како да започнеш со cloud?

- Евалуација на бизнис случај за public, private и hybrid cloud
- Развој на стратегија за интеграција и миграција
- Преглед на апликации за SaaS кандидати
- Планирање на безбедност, управување и мониторинг1.

---

11. Практични примери за користење на cloud во ентерпрајз

- Hybrid cloud за скалабилност и контрола
- Test/Development платформи
- Disaster Recovery
- Cloud file storage за backup
- Load balancing за peak usage
- Намалување на overhead, подобра контрола на мрежа и трошоци, алтернативи за messaging, брза имплементација1.

---

12. Важни предизвици за безбедност и приватност

- Голем trusted computing base (TCB) кај виртуелизацијата – што е поголем, толку е потешко да се обезбеди целосна безбедност1.
- Потребни се јасни политики за пристап, backup, мониторинг и одговор на инциденти1.

Презентација 8:

Основи на интеграција

- Интеграцијата подразбира поврзување на два или повеќе системи, кои може да бидат на различни технологии, платформи и локации, со цел размена и обработка на податоци.
- Рачниот трансфер на податоци е подложен на грешки и бара човечки ресурси.
- Интеграцијата не е само за комуникација, туку и за разбирање и обработка на содржината на податоците1.

Планирање на интеграција

- Потребно е прецизно дефинирање на барањата за интеграција.
- Опис на интерфејсите на системите што се интегрираат.
- Можност за тест интерфејс.
- Организација на дозволи за пристап до системите1.

Најчести методи за трансфер на податоци

- Web services (HTTP/HTTPS)
- FTP (File Transfer Protocol)
- SOAP (XML-базиран протокол)
- REST (архитектонски стил, често со JSON/XML)
- GraphQL (флексибилно структурирање на податоци)
- Други: JSON-RPC, gRPC, Thrift1.

Протокол	**Формат**	**Клучна предност**
SOAP	XML	Стабилност, стандардизација
REST	JSON/XML и др.	Флексибилност
gRPC	Protocol Buffers/JSON	Моќна дефиниција на функции
GraphQL	JSON	Флексибилно структурирање

Протокол	Формат	Клучна предност
Thrift	JSON/Binary	Адаптабилност

Типови на формати

- JSON: Лесен за употреба, широко прифатен.
- XML: За комплексни структури и метаподатоци.
- Binary: За ефикасен трансфер, но помалку читлив1.

Приоди кон интеграција

- XML интеграција: Универзален формат за размена на податоци, овозможува „преведување“ меѓу различни системи.
- SOA/Web Services: Сервисно-ориентирана архитектура, повторна употреба на компоненти преку стандардизирани интерфејси.
- COTS (Commercial-off-the-shelf): Купување готови софтверски решенија, со ограничена можност за прилагодување.
- Cloud-based: Користење облачни сервиси, скалабилност и флексибилност1.

COTS интеграција

- Брзо имплементирање, намалени трошоци и време за развој.
- Ограничена можност за прилагодување според специфични барања1.

SOA (Service-Oriented Architecture)

- Овозможува повторна употреба и интероперабилност на компоненти.
- Сервисите се дефинираат преку интерфејси, овозможувајќи лесно поврзување и намалување на зависности.
- Придобивки: поголема агилност, повторна употреба, интеграција на легаси системи, подобра соработка меѓу бизнис и ИТ1.

SOA vs Microservices

SOA	Microservices
Архитектура на интеграција	Апликациска архитектура
Крупни, повторно употребливи сервиси	Мали, независни сервиси
Фокус на интеграција на системи	Фокус на скалабилност и одржување
Loosely coupled, но на повисоко ниво	Fine-grained, независно распоредливи

ESB (Enterprise Service Bus)

- Софтверска архитектура што поврзува повеќе системи преку централизирана „бус“ инфраструктура.
- Овозможува поврзување без прекини и лесно додавање нови системи1.

Cloud Native пристап

- Апликации дизајнирани за скалабилност, брзи промени и толеранција на грешки.
- Користат микросервиси, контејнери, DevOps и континуирана испорака.
- Предности: брз развој, скалабилност, подобра безбедност.
- Недостатоци: зголемена комплексност, потреба од нови алатки и култура на организацијата1.

Клучни поими и концепти

- Tight vs Loose Coupling: Тесно поврзани системи се поефикасни, но помалку флексибилни; лабаво поврзани се пофлексибилни, но со помалку ефикасна комуникација.
- Компонентен модел vs Web Services: Компонентите се наменети за внатрешна интеграција, web services за надворешна и меѓуорганизациска интеграција.
- Составување на сервиси: Поврзување на повеќе сервиси за создавање комплексни бизнис процеси1.

Што треба да запомниш:

- Интеграцијата е клучна за модерните ИТ системи и бара внимателно планирање.
- Постојат различни технологии и архитектури за интеграција: XML, SOA, COTS, Cloud-native.
- SOA и микросервиси се два различни пристапи со различни предности и примени.
- ESB и cloud-native пристапите овозможуваат скалабилност и флексибилност.
- Изборот на технологија зависи од бизнис барањата, постојната инфраструктура и очекуваната динамика на промените1.

Презентација 9:

1. Интеграција на безбедност во SDLC (Software Development Life Cycle)

- **SDLC** модели (Waterfall, Agile, DevOps, DevSecOps) се користат за планирање, извршување и контрола на развојот на софтвер, при што безбедноста треба да се интегрира на секој чекор.
- SDLC фазите според NIST: иницирање, развој/аквизиција, имплементација, операција/одржување, отстранување (disposal)1.
- Секоја фаза има свои безбедносни активности: од дефинирање на барања и анализа на ризик, преку тестирање и сертификација, до одржување и отстранување1.

---

2. Модели на развој на софтвер

- **Waterfall**: Секоја фаза мора да заврши пред следната; повеќе формален и безбеден, но не-флексибилен.
- **Spiral**: Фокус на управување со ризици, со редовни ревизии.
- **Agile/Scrum/XP**: Флексибилни, брзи, со честа комуникација и инкрементални испораки.
- **DevOps/DevSecOps**: Континуирана интеграција и испорака, со безбедност интегрирана во секоја фаза1.
- **Prototyping, RAD, JAD, Component-based, Reuse, Clean Room**: Различни пристапи за побрз развој, подобра интеракција со корисници, или фокус на квалитет1.

---

3. Модели на зрелост (Maturity Models)

- **CMM, CMMI, BSIMM, SAMM**: Користени за мерење и подобрување на процесите за развој и безбедност на софтверот. Овозможуваат идентификација на слабости и поставување стандарди1.

---

4. Интегрирани тимови и процеси

- **Integrated Product Team (IPT)**: Мултидисциплинарни тимови за комплексни проекти, со фокус на заедничко носење одлуки.
- **Integrated Product and Process Development (IPPD)**: Комбинирање на дизајн на производ и процес за подобро исполнување на барањата1.

---

5. Безбедносни контроли во екосистемот за развој

- **Контроли**: Програмирање, библиотеки, алатки, IDE, runtime, CI/CD, SCM, code repositories, тестирање (SAST/DAST), SOAR.
- **Тестирање**: White Box (структурно), Black Box (функционално), статичко/динамичко, traceability matrix, различни нивоа на тестирање (unit, integration, system, acceptance)1.

---

6. Објектно-ориентирано програмирање и безбедност

- Клучни концепти: енкапсулација, полиморфизам, наследување, кохезија и coupling.
- Висока кохезија и ниско coupling се пожелни за полесно одржување и безбедност1.

---

7. Управување со трети страни и отворен код

- **Трети страни и open source**: До 90% од апликацијата може да биде составена од компоненти од трети страни. Овие компоненти често носат скриени или познати ранливости (CVEs).
- **Транзитивни зависности**: Зависности во зависностите, што ја зголемува комплексноста и ризикот.

- **Software Escrow**: Неутрална трета страна го чува изворниот код за заштита на купувачот1.

---

8. Проценка и управување со ризици

- **Risk Analysis**: Идентификација и приоритизација на ризици, квантитативна и квалитативна анализа.
- **Risk Mitigation**: Прифаќање, избегнување или контрола на ризикот, contingency plans.
- **Auditing and Logging**: Следење и документирање на промени, за подобра безбедност и одговорност1.

---

9. Современи закани и ранливости

- **Buffer Overflow**: Една од најстарите и најчести ранливости, често поради лошо проверување на граници.
- **SQL Injection, XSS, Session Hijacking**: Чести веб-апликациски напади.
- **Covert Channels, Memory/Object Reuse, Social Engineering, TOC/TOU**: Различни типови на напади и слабости во софтверот.
- **Web Threats**: Drive-by downloads, clickjacking, malvertising, трети страни, мобилни апликации1.

---

10. Безбедносни практики и стандарди

- **Secure Coding Guidelines**: Практики за безбедно програмирање, заштита на API, software defined security.
- **Меѓународни стандарди**: ISO/IEC 27034, ISO 12207, NIST SP800-64 и други, за безбедност во SDLC.
- **Контрола и одвојување на околини**: Development, QA/Staging, Production, со физичка и логичка изолација, ACL, separation of duties1.

---

11. Тестирање, сертификација и акредитација

- **Тестирање**: Статичко/динамичко, различни нивоа и типови.
- **Сертификација/акредитација**: Формален процес за потврда на безбедноста, со континуирано следење и ажурирање1.

---

12. Клучни поенти

- Безбедноста мора да биде интегрирана во секоја фаза од развојот на софтверот.
- Управувањето со ризици, тестирање и контрола на трети страни се критични за безбедна имплементација.
- Современите закани се динамични и бараат континуирана едукација и ажурирање на практиките.
- Примената на стандарди и модели на зрелост помага да се одржи висок квалитет и безбедност.