Въведение в трислойния модел

Client – Services – Database

Какво е трислоен модел?

- Разпределя приложението на слоеве
- Всеки слой има строго определена задача
- Във Visual Studio можем да създадем такова приложение създавайки различни проекти в рамките на Solution-а ни

Какво е трислоен модел? (2)

Слой за данни /Data Access Layer/

- Отговаря за връзка с БД
- Съхранява данните
- Изпълнява заявки и команди върху БД
- Не позволява данните да бъдат достъпвани и манипулирани директно от клиента в презентационния слой
- Предоставя възможност за управление на информацията без значение от съхраняващия механизъм
- Носи ползи за мащабируемостта и поддръжката на приложението

Слой за услуги

- Наричан още: бизнес слой, логически слой /business logic layer, service layer, middle layer, logic layer/
- Отговаря за:
  - Обработка на данните приети от презентационния слой
  - Заявяване на данни от слоя за данни
  - Обработка на получените данни от слоя за данни
  - Подготовка на данните за презентационния слой

Презентационен слой

- Отговаря за:
  - Въвеждане на данни от потребителите
  - Визуализиране на данни към потребителите
- Може да бъде:
  - Графичен потребителски интерфейс
  - Уеб приложение, мобилно приложение и др.

1. Въведение и цели на курса

- Основна идея: курсът свързва всички знания, натрупани до момента, в един завършен софтуерен проект.
- Цели: разбиране на структурата на приложенията, работа с база данни, интерфейси, тестове и добри практики.
- Подчертаване на три основни умения: **мислене за структура**, **работа в екип**, **качествен код**.

2. Структура на софтуерен проект – трислоен модел

- **Слой за данни (Data Access Layer):** връзка с база данни, SQL заявки, ORM.
- **Слой за услуги (Service Layer):** бизнес логика, обработка на данни.
- **Слой за потребителски интерфейс (UI Layer):** визуализация – конзола, desktop, web, mobile.
- Пример: CRUD приложение с три ясно отделени слоя.

3. Тестване, дебъгване и рефакториране

- **Unit тестове:** покриване на кода с автоматизирани проверки.
- **Регресия:** предотвратяване на стари грешки, които вече са били поправени – обикновено след промени в кода
- **Дебъгване:** използване на инструменти за откриване и поправяне на дефекти и непредвидени резултати.
- **Рефакториране:** пренаписване на код, като се гарантира четимост, тестеруемост и гъвкавост.
- **Добри практики:** тестове + малки стъпки при промени.

4. Инструменти и външни библиотеки

- IDE и текстови редактори – разлики, разширения, клавишни комбинации.
- Управление на пакети (package managers) – npm, pip, NuGet и др.
- Външни библиотеки: как да четем документация, какво печелим от тях.

5. Работа с бази данни и ORM

- Свързване на приложение с база.
- SQL заявки през програмен език.
- CRUD операции (Create, Read, Update, Delete).
- ORM – по-лесна работа чрез обектно-релационно съответствие.
- Пример: проста програма за управление на потребители.

6. Приложения с различни интерфейси

- Защо е важно отделянето на бизнес логиката от UI.
- Един и същ „сървис“ слой може да се използва от конзолен интерфейс, web приложение и мобилно приложение.
- Пример: библиотека за услуги, която се използва от два различни фронта.

7. Курсов проект и екипна работа

- Крайната цел: **трислойно приложение** с поне два интерфейса и база данни.
- Включва: външни библиотеки, unit тестове, спазване на style guide.
- Практическа полза: симулация на реална работа в екип.