https://www.halvorsen.blog

Week Assignment

Software Architecture

Hans-Petter Halvorsen

Week Assignment

1. Get an overview of different types of **Architecture** in general
   - Client-Server, APIs, 3-Layer Architecture, MVC, Web Services and REST APIs, Web Architecture
2. Continue **Implementing your System**
   - Make sure to select, use and implement a proper Architecture for your system
   - Work with "Beta" Iteration (Iteration #2)
   - Make sure your Architecture is documented in the Software Requirements and Design (SRD) document – Make one or more System Sketches and sketches of the Architecture of the system
3. Do **Scrum**
   - Have Scrum Meetings and use Scrum features in Azure DevOps

https://www.halvorsen.blog

Software Architecture

Hans-Petter Halvorsen

Table of Contents

Software Architecture

- What is Software Architecture?
- Client-Server Architecture
- APIs and Libraries
- 3 Layer/Tier Architecture (N Tier Architecture)
- Model-View-Controller (MVC)
- Web Services (Service Oriented Architecture, SOA)
  - REST API

What is Software Architecture?

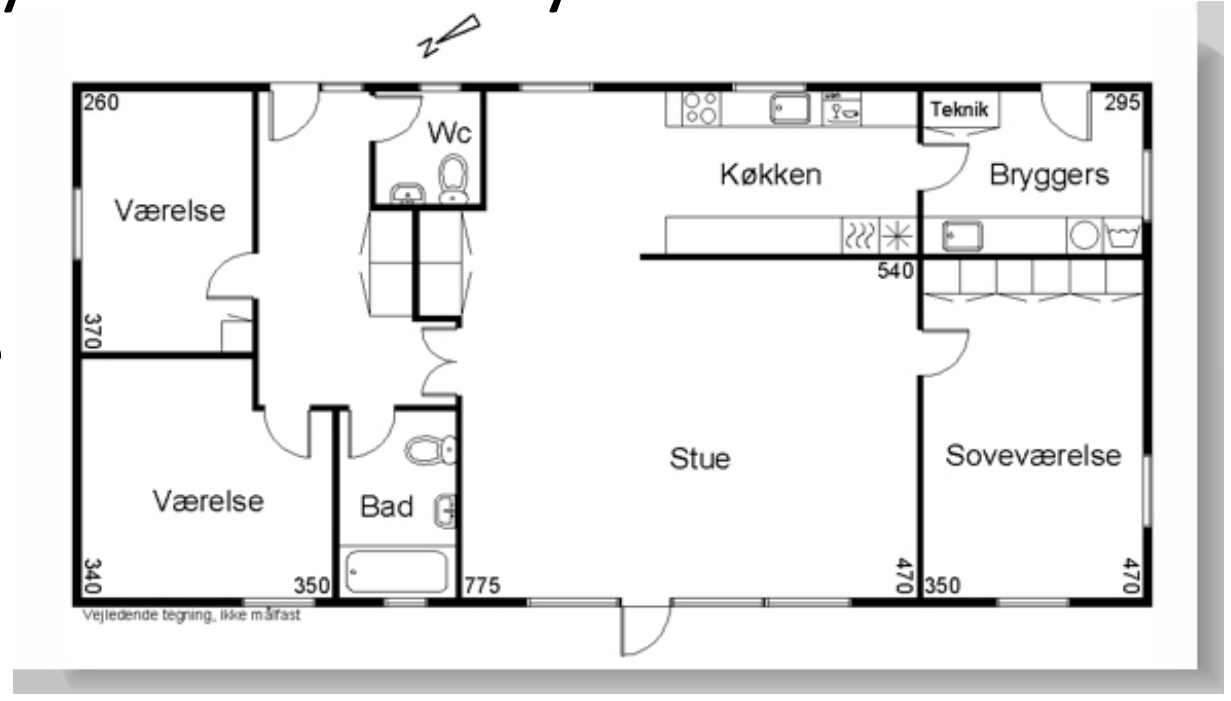
Software architecture is the high level structure of a software system, the discipline of creating such structures, the documentation and Implementation of these structures

http://en.wikipedia.org/wiki/Software_architecture

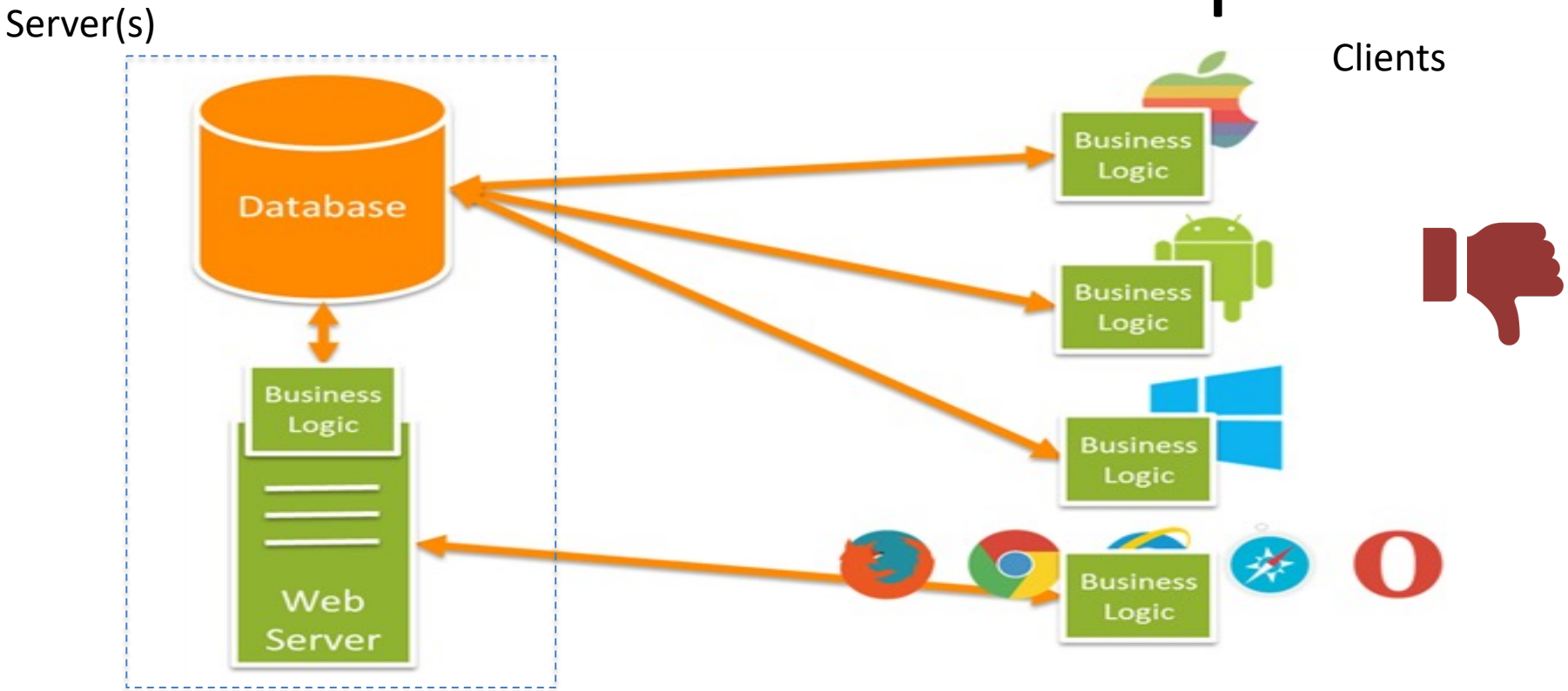
What is Software Architecture?

-> A “model” of your Software System

Important to have when “building” your Software

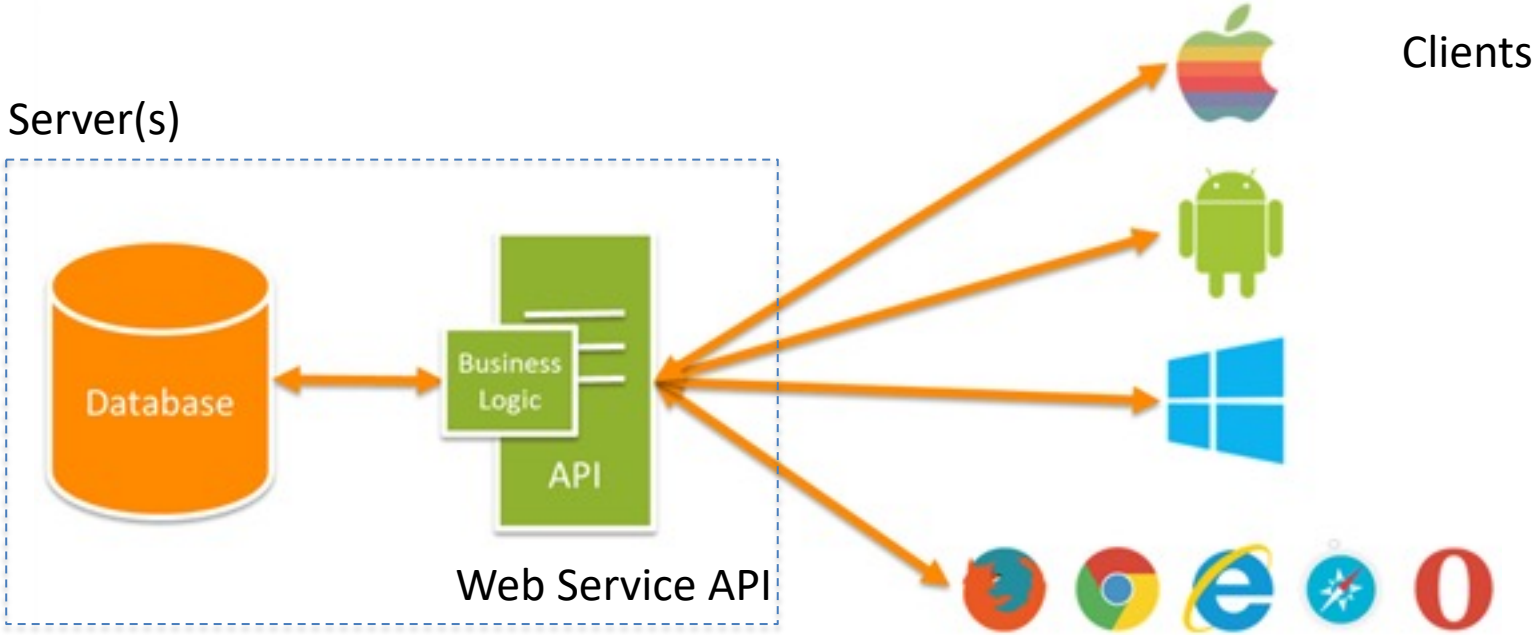


Bad Architecture Example



Let's consider a modern application which may include several mobile apps on various platforms and usually some kind of web application too. Without an API, a basic architecture may look like this where each client app has its own embedded business logic.

Good Architecture Example



Each app uses the same API to get, update and manipulate data. All apps have feature parity and when you need to make a change you just make it in one place in line with the 'Don't Repeat Yourself' (**DRY**) principle of software development. The apps themselves then become relatively lightweight UI layers.

Software Architecture Styles

2-Tier

Client-Server

3-Tier: A way to structure your code into logical parts. Different devices or software modules can share the same code.

Web Services

3-Tier Architecture

n-Tier

Good Software!

APIs & Libraries

Web Services: A standard way to get data over a network/Internet using standard Web protocols (HTTP, etc.)

There are many others, but we will focus on these

API: Application Programming Interface. Different devices or software modules can share the same code. Code once, use it many times.

– Det gjenstår en del testing i samarbeid med Difi som oppdragsgiver.

Det viste seg at løsningen ikke holdt mål, hverken i forhold til de formelle kravene fra Difi eller det øvrige markedet. Derfor måtte selve kjerne-teknologien bygges om for å kunne håndtere mer komplekse oppgaver.

"Rent teknisk har vi bygget om til en lagdelt arkitektur, slik at E-boks kan utvikles på en mer moderne plattform. Det skal gi bedre forhold på både sikkerhet, kapasitet og fleksibilitet"

E-boks har tapt kampen mot klokka etter at det ble nødvendig å bygge helt ny løsning i innspurten av kontrakten om sikker digital postkasse. Illustrasjonsfoto: Colourbox

http://www.digi.no/itnorge/2015/03/04/mer-forsinkelser-for-e-boks

DIGITAL POSTKASSE

Mer forsinkelser for E-boks

Måtte bygge om hele plattformen i innspurten.

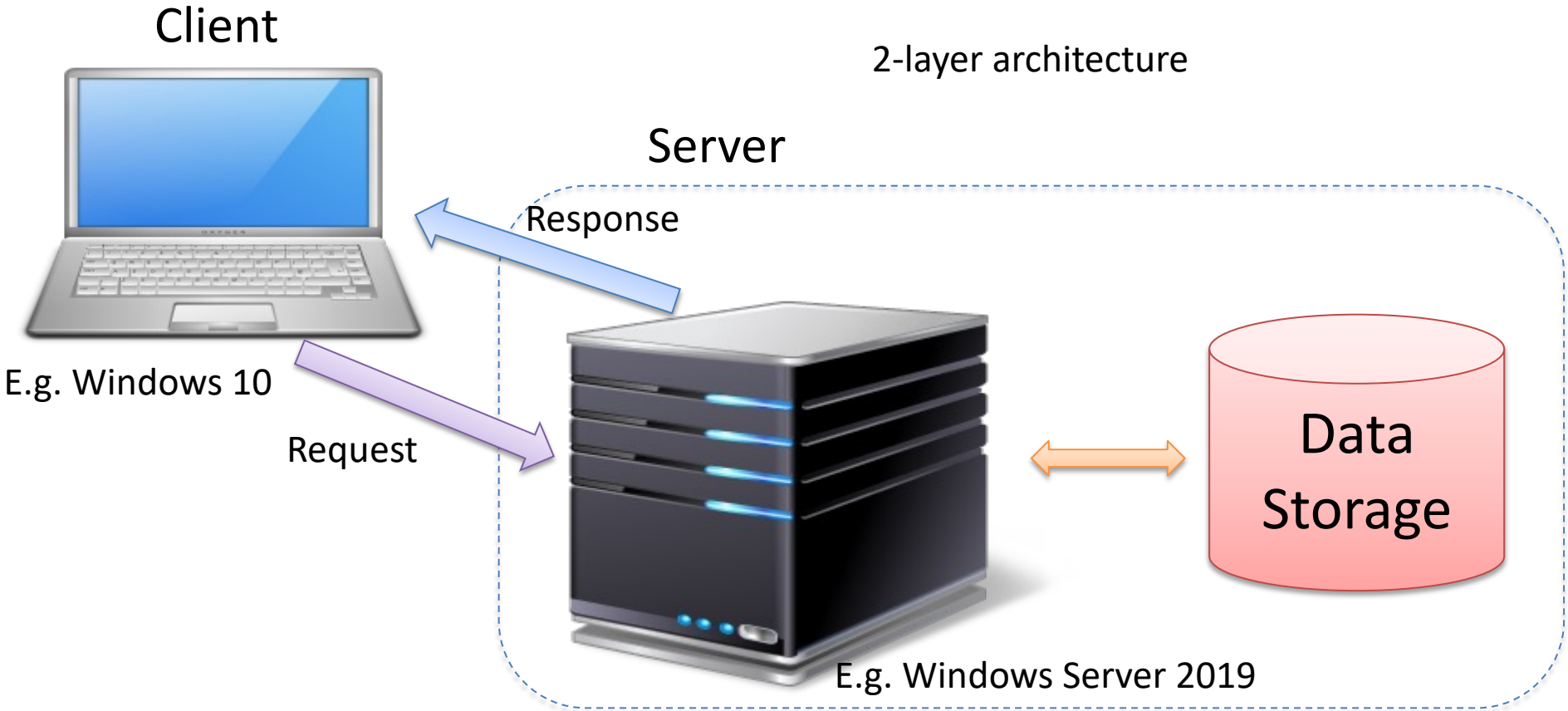
https://www.halvorsen.blog

Client-Server Architecture

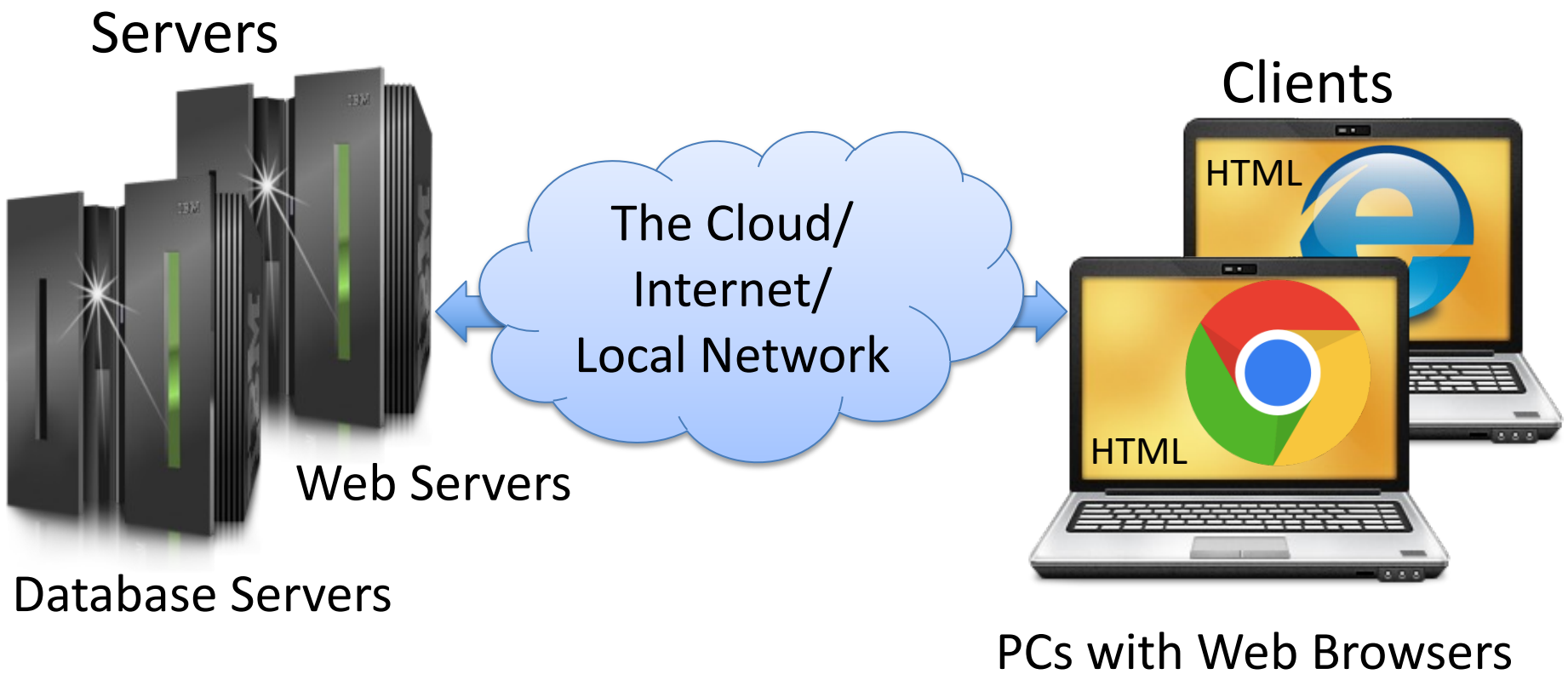
Hans-Petter Halvorsen

Table of Contents

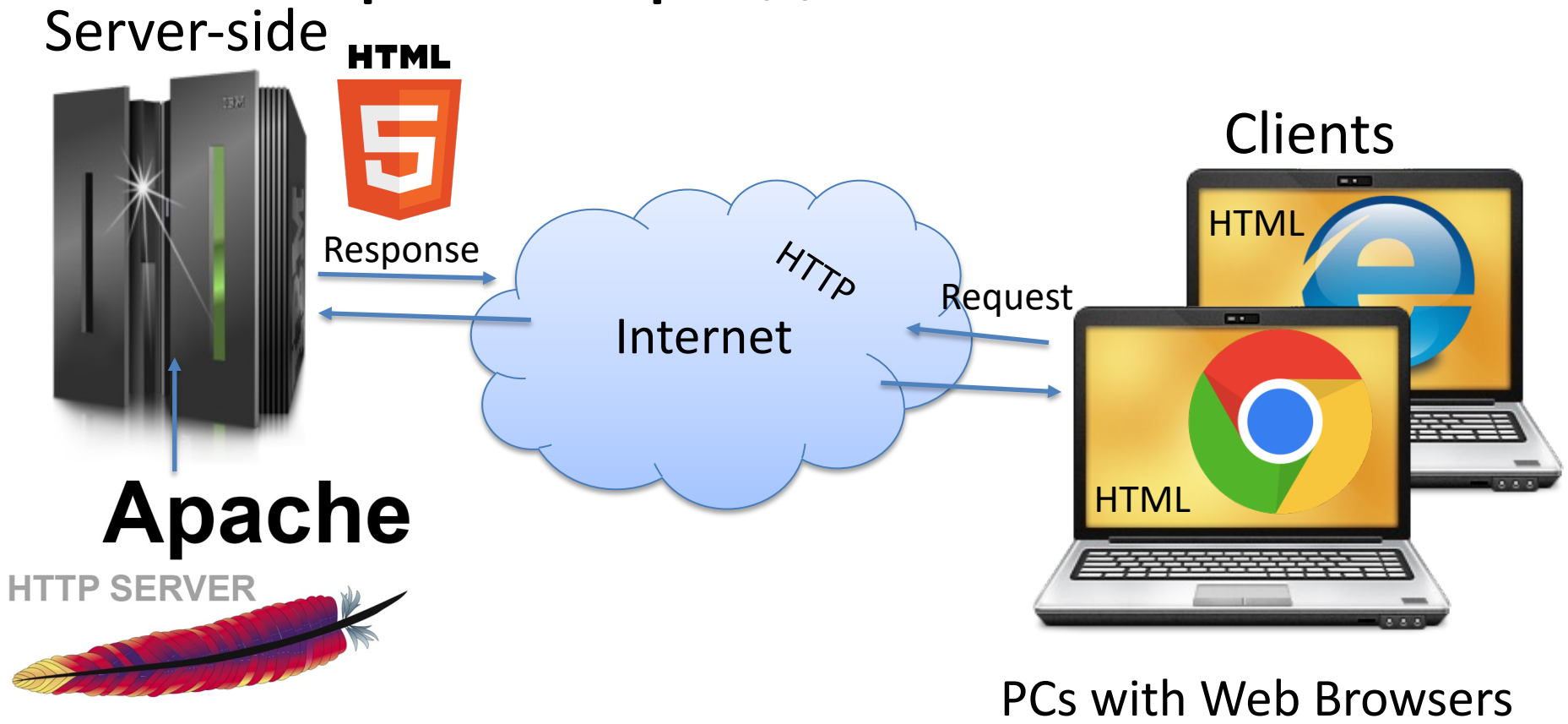
Client-Server



Web Architecture



Example: https://web01.usn.no



APIs

Hans-Petter Halvorsen

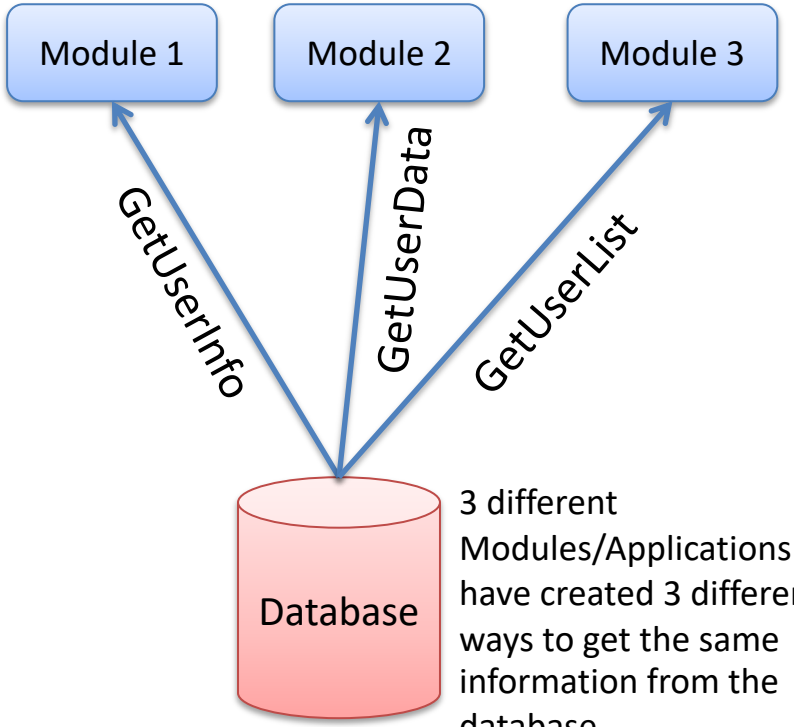
Table of Contents

API/Library

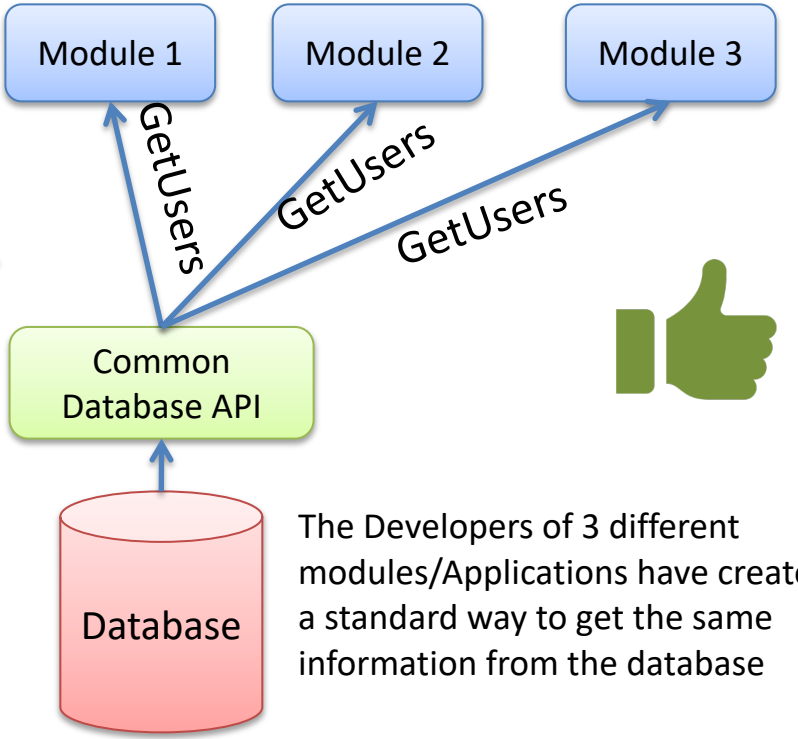
- API - Application Programming Interface
- A specification of how some software components should interact with each other.
- A library with functions, etc. you can use in your code
- Examples:
  - Windows API
  - Java API
  - ADO.NET ...
- But you can also create your own API that you use internally in the development team or expose to others

API Example

Database API



3 different Modules/Applications have created 3 different ways to get the same information from the database

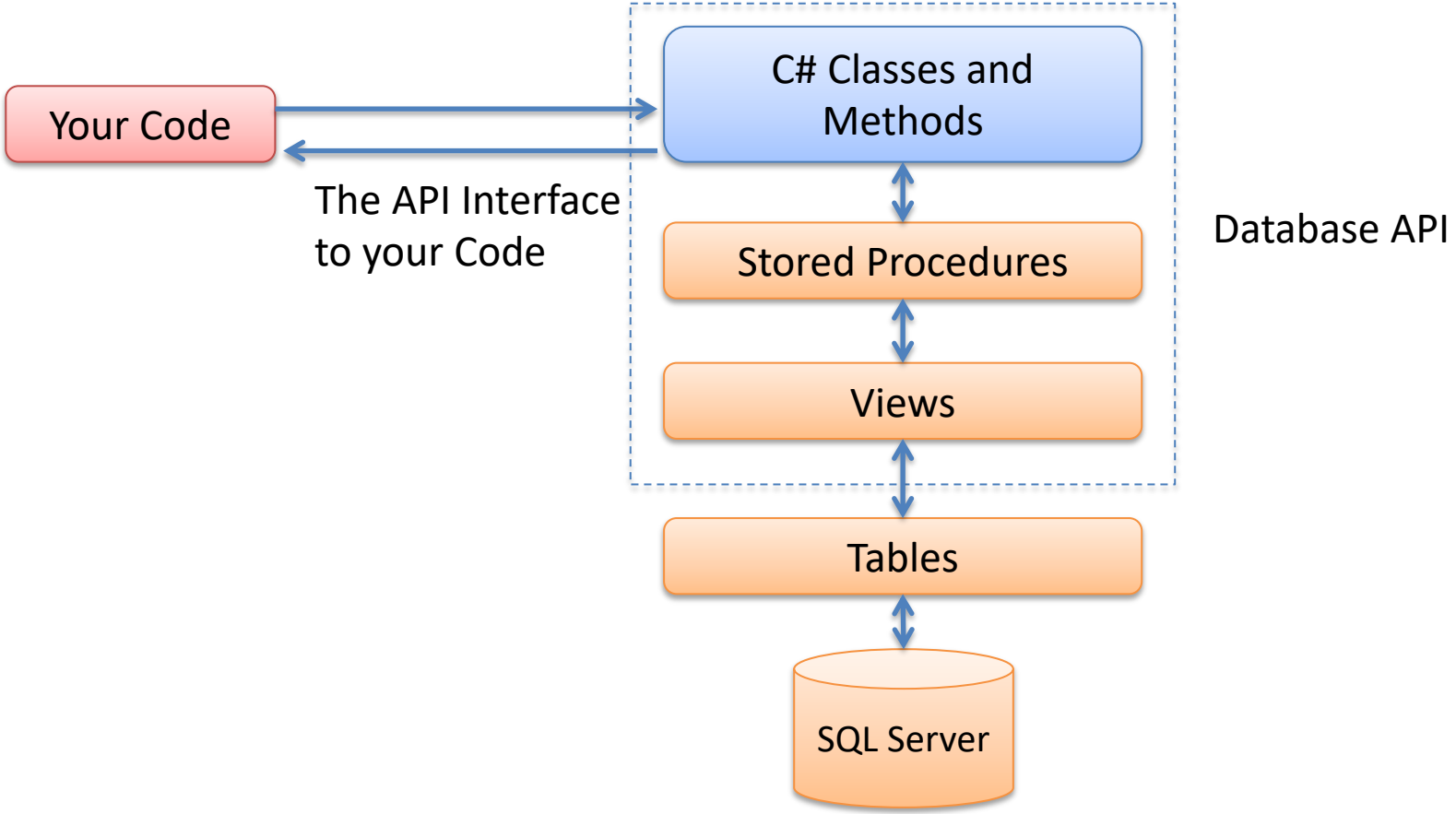


The Developers of 3 different modules/Applications have created a standard way to get the same information from the database

Database API

- Database Communication API
- Stored Procedures and Views as a natural part of such a Database API.
- API using Layers:
  - A good practice is to create a New Project with one or more Classes/Methods (Class Library) where you put the API code that can be shared among the Developers.
  - Each Developer can then add this Project to their Solution in order to use it and maintain it.

A Typical Database API



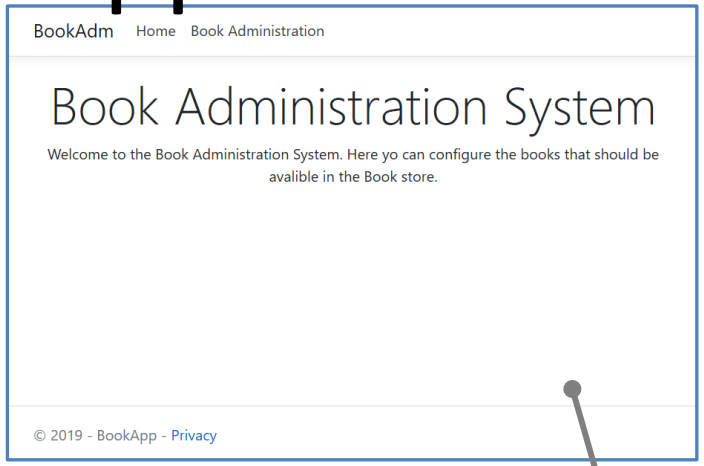
The Benefits of API Driven Design

When an API is used in a project, it

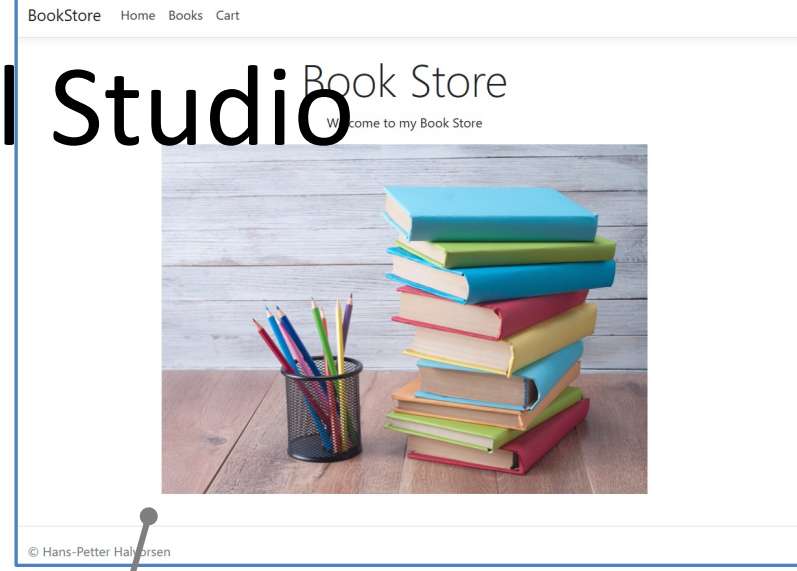
- Allows to focus on the project
- Saves development time
- Reduces errors and debugging
- Facilitates modular design
- Provides a consistent development platform

➔ API driven design requires planning and programming skills. API driven design is costly initially, but it pays in the long run. So, obviously, creating APIs is good software practice in most cases.

2 Applications in Visual Studio



Application #1



Application #2

C# Class

GetBooks()

GetBookInformation()

etc.

C# Class

GetBooks()

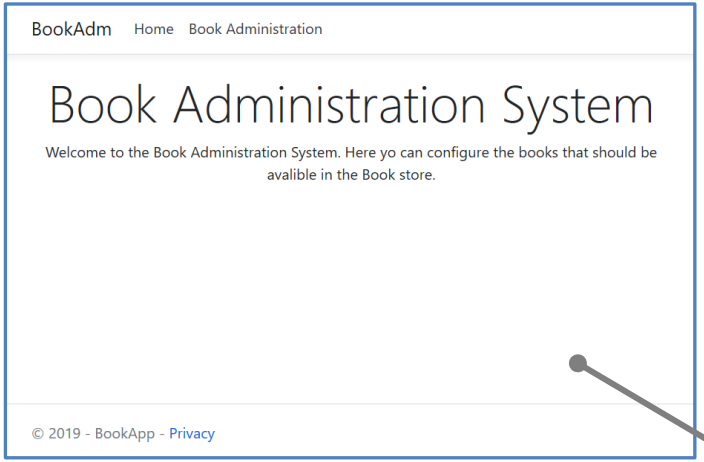
GetBookInformation()

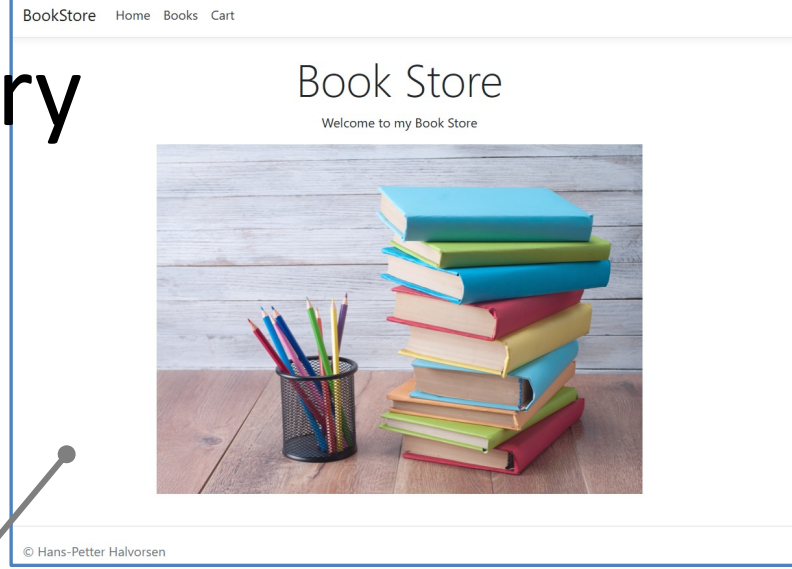
etc.

Database

Using Class Library

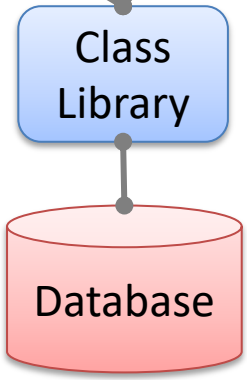
Application #1





Application #2

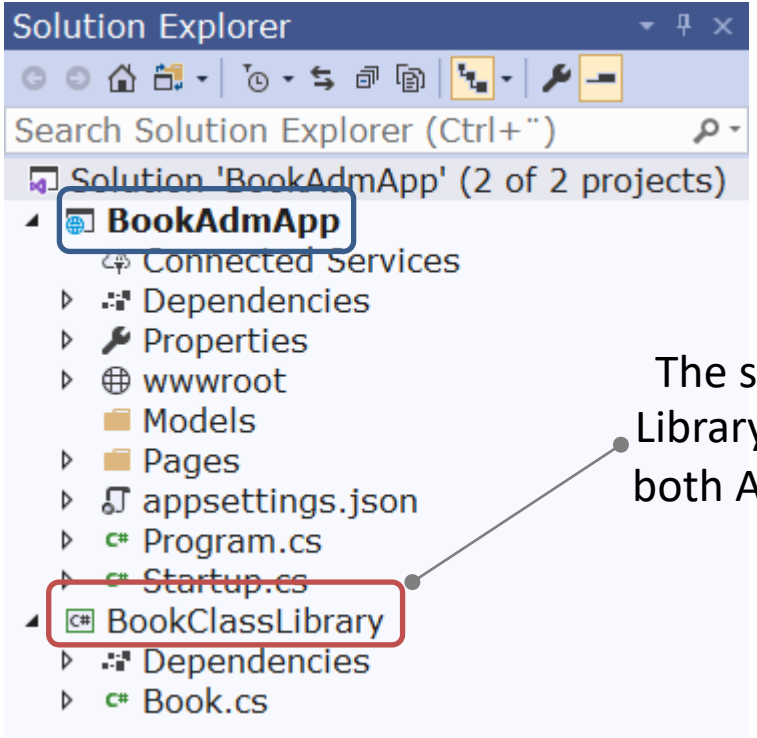
Put everything that is common for these 2 Application into a Class Library



GetBooks()
GetBookInformation()
etc.

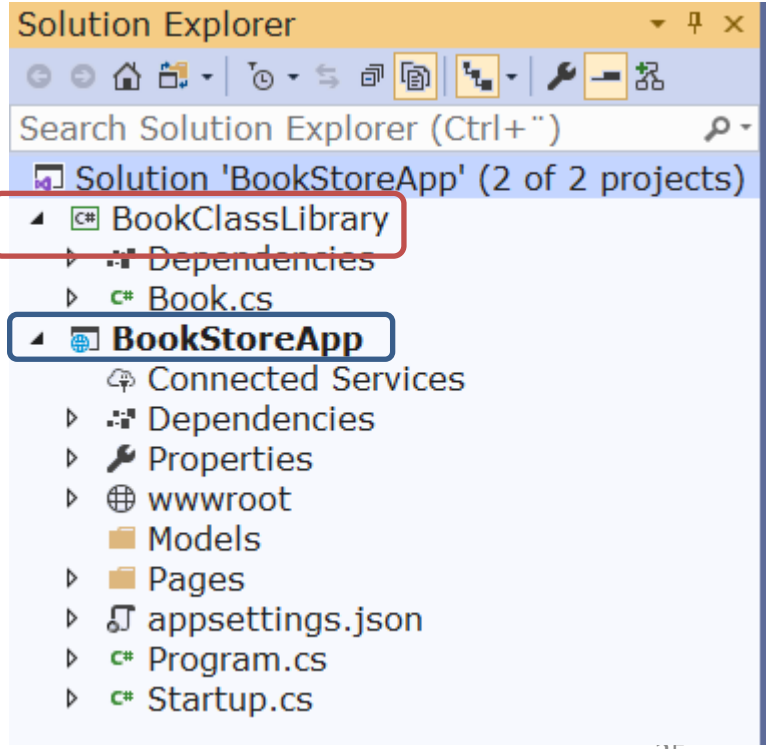
Class Library Example - Solution Explorers

Application #1



The same Class Library is used by both Applications

Application #2



API Summary

Use of an API will make the architecture of your application much cleaner, making it easier to add features and fix bugs as your project progresses.

https://www.halvorsen.blog

3 Layer/Tier Architecture

Hans-Petter Halvorsen

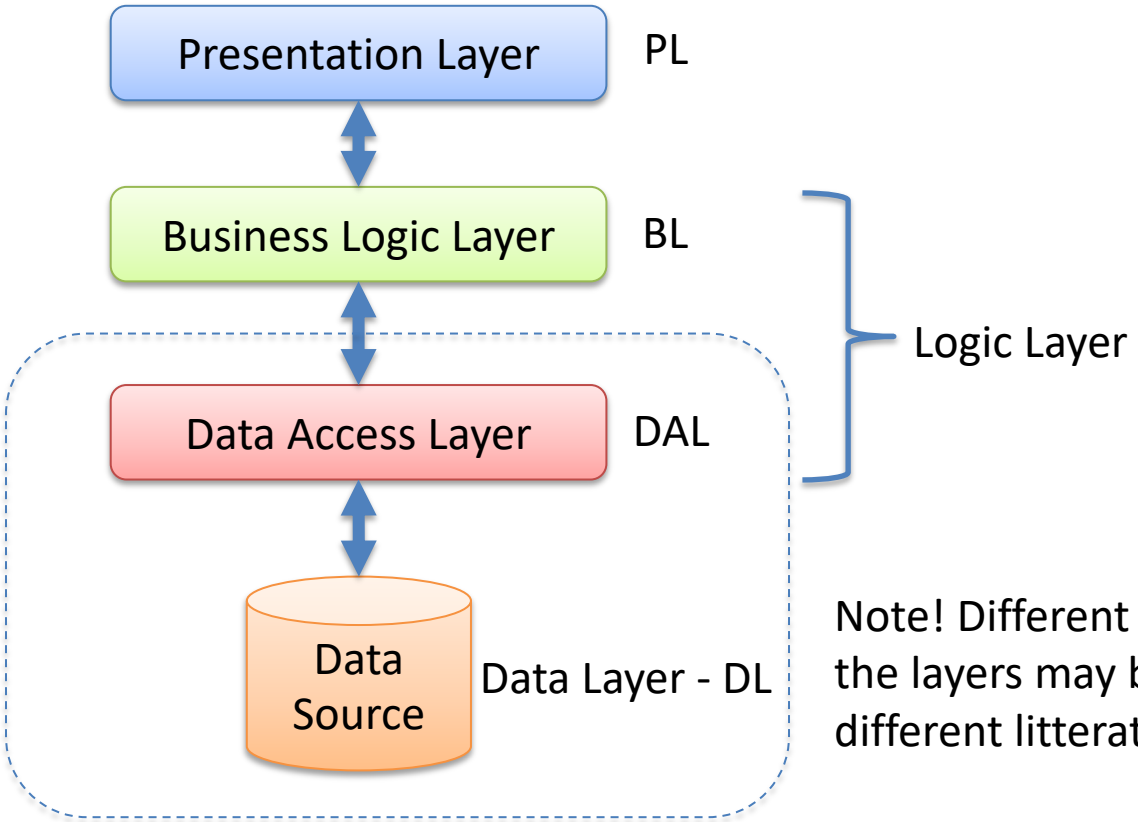
Table of Contents

Layers vs. Tiers

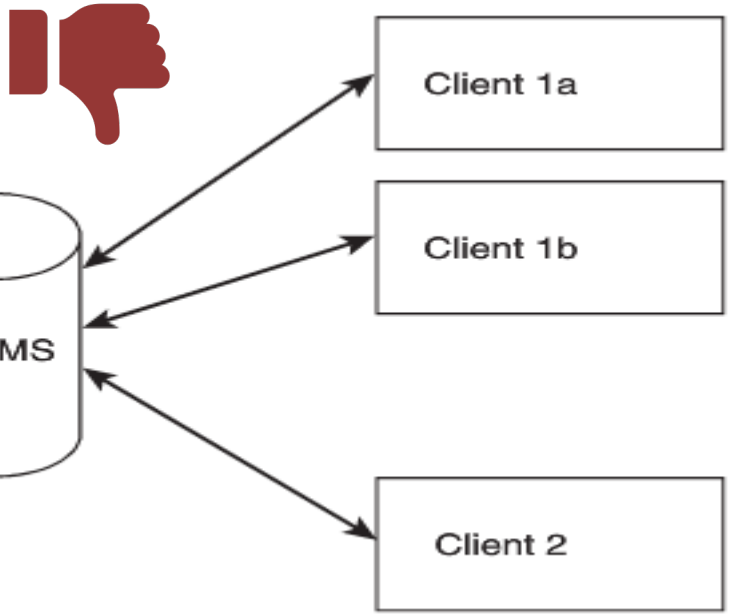
- “Layers” are a logical separation and “Tiers” are a physical separation
- The Terms are often mixed
- It is probably better to use the terms “Logic Layer” and “Physical Layer” if you need inform that they are on the same computer or not

3 Tier/Layer Architecture

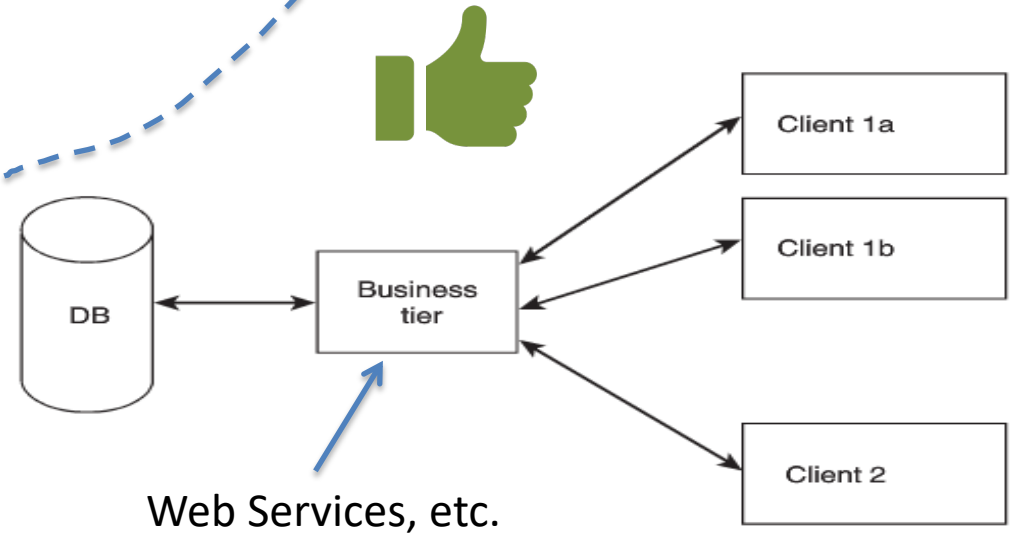
Note! The different layers can be on the same computer (Logic Layers) or on different Computers in a network (Physical Layers)



Note! Different Names for the layers may be used in different litterature



2-tier: The database-centric style. Typically, the clients communicate directly with the database.

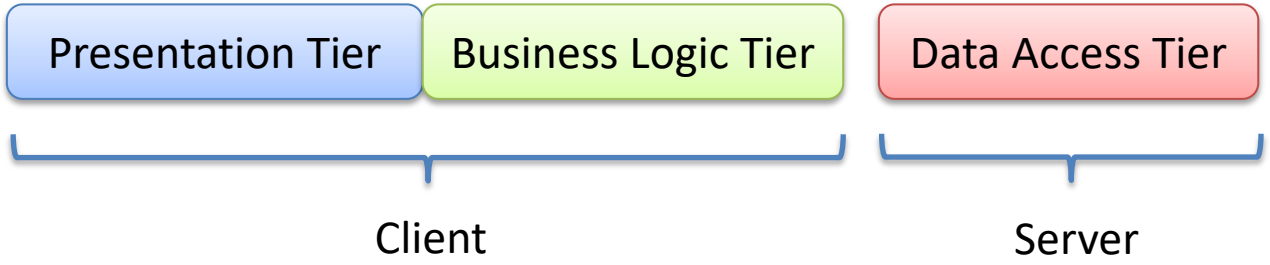


A three-tier style, in which clients do not connect directly to the database.

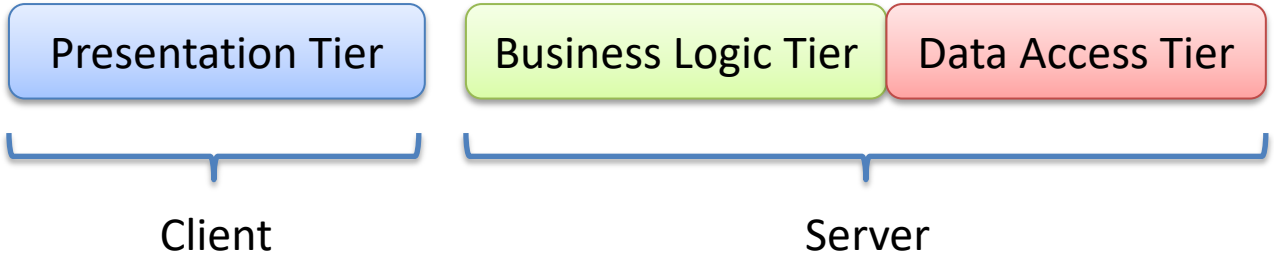
Web Services, etc.

Fat Client vs. Thin Client

Fat Client:



Thin Client:



3 Tier Application (Physical Layers)

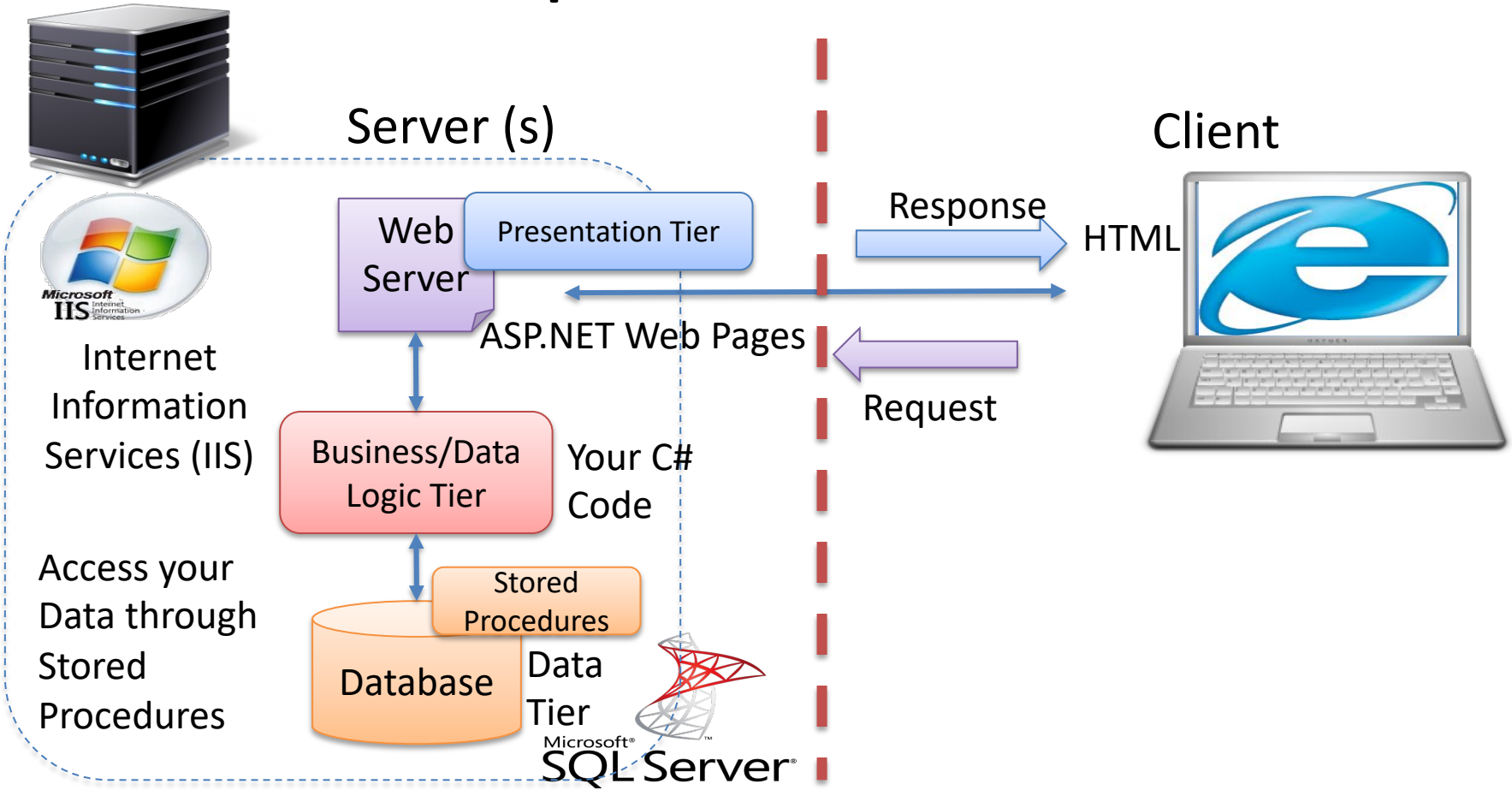
Client Program

Application Server

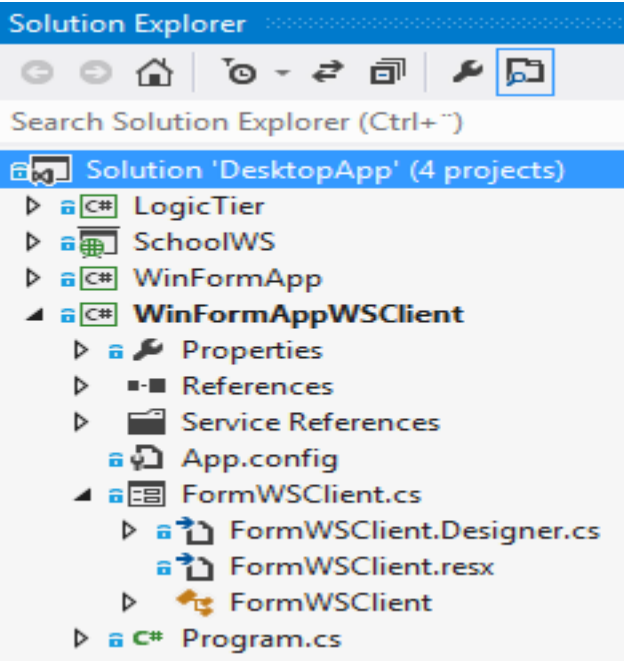
Database Server

Note! The different layers can be on the same computer (Logic Layers) or on different Computers in a network (Physical Layers->Tiers)

3-tier Example – ASP.NET Web Site

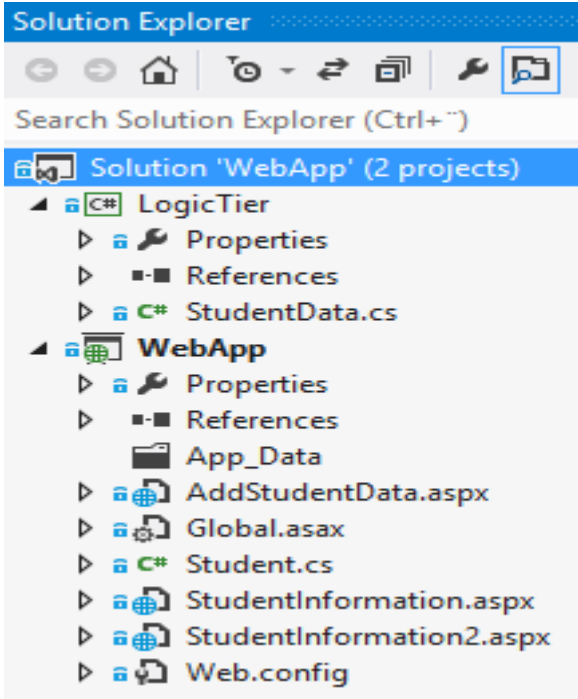


3-layer (Logical) Examples in Visual Studio

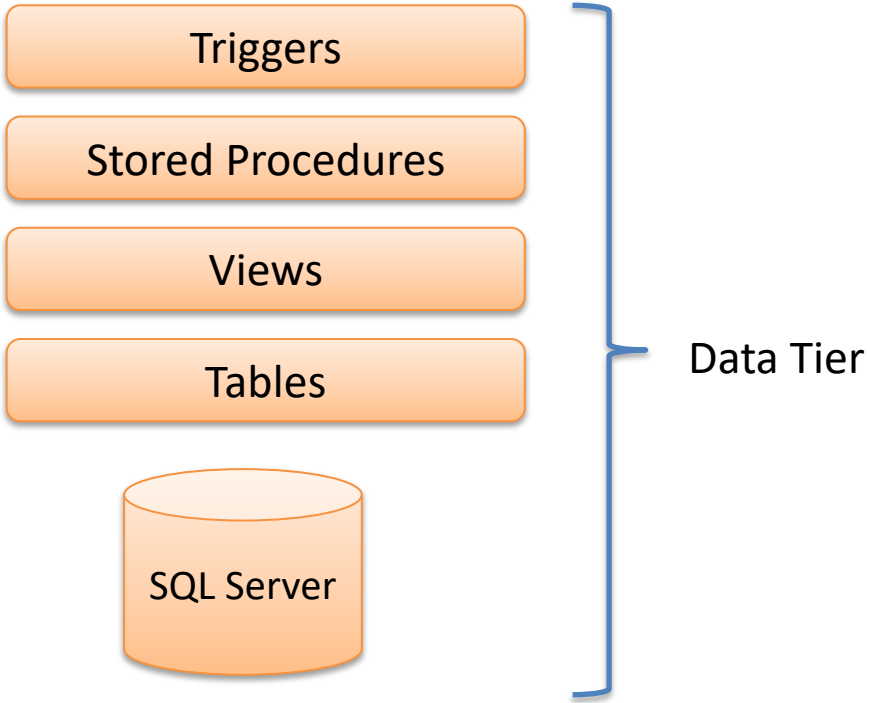


- Your Solution in Visual Studio may contain one or more Projects.
- Each Project will then be a Layer.
- The same Project can then be used in several Solutions, meaning several Developers can maintain and use the same Project (Layer)

This is something you should consider doing in your project



Data Tier

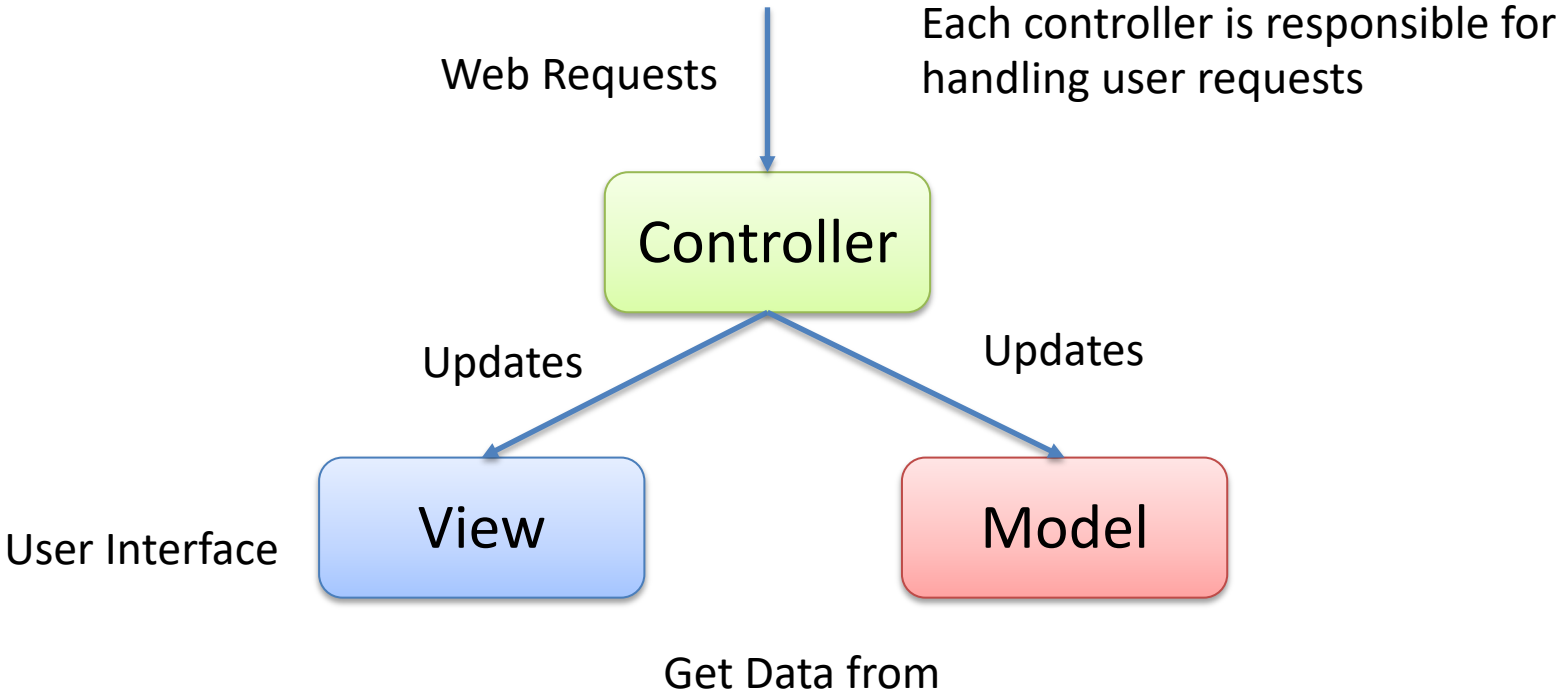

Triggers
Stored Procedures
Views
Tables
SQL Server
Data Tier

MVC

Hans-Petter Halvorsen

Table of Contents

MVC



Note! In ASP.NET Core it is possible to use the MVC architecture

Model-View-Controller (MVC)

The Model-View-Controller (MVC) architectural pattern separates an app into three main components:

- **Models (M):** Classes that represent the data of the app. Typically, model objects retrieve and store data into a database.
- **Views (V):** Views are the components that display the app's user interface (UI). Generally, this UI displays the model data.
- **Controllers (C):** Classes that handle browser requests, retrieve model data, and then specify view templates that return a response to the browser.

In an MVC app, the view only displays information; the controller handles and responds to user input and interaction. For example, the controller handles route data and query-string values and passes these values to the model. The model might use these values to query the database.

3. Layer vs. MVC

ASP.NET

ASP.NET supports three different development models:

ASP.NET Razor

The recommended approach

ASP.NET MVC

For more experienced Web Developers

ASP.NET Web Forms

The Web version of classic WinForms. GUI and Code is separated. If you are familiar with WinForms or WPF, this is a good choice

https://www.halvorsen.blog

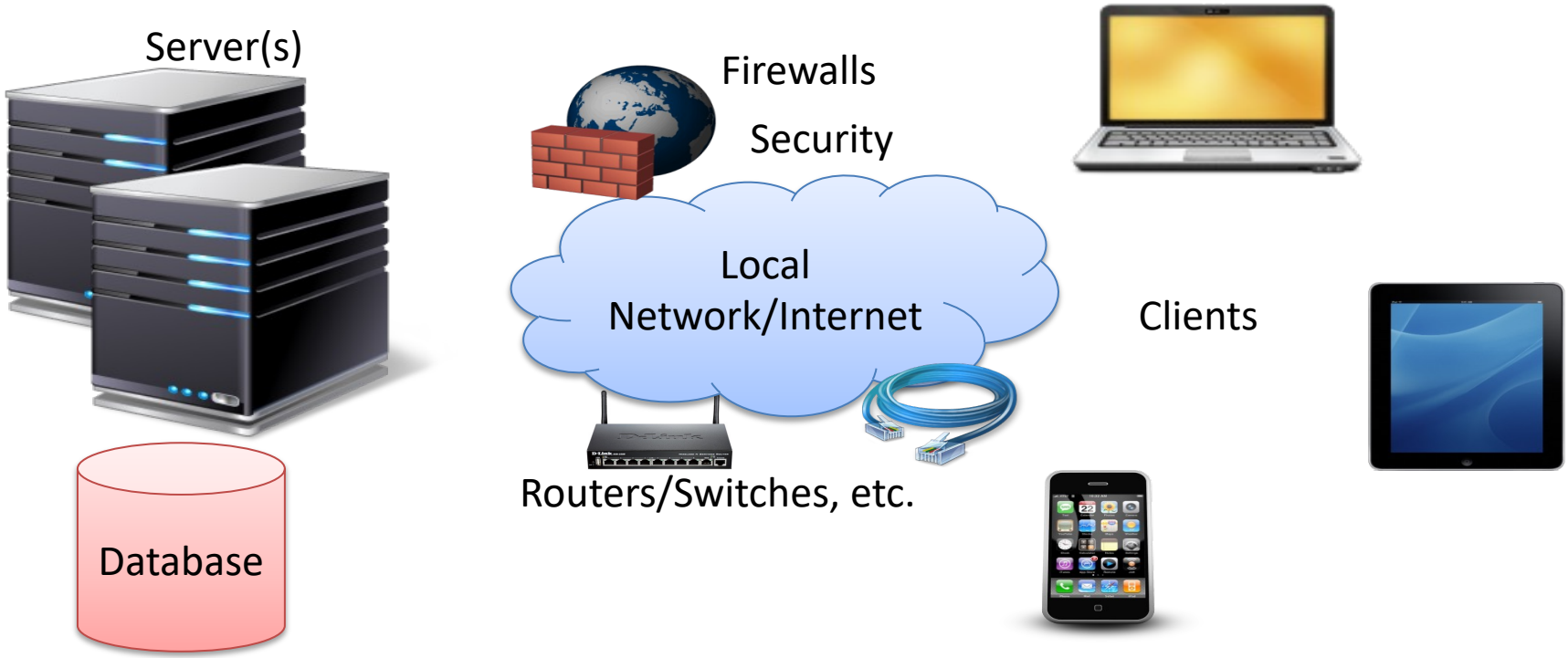
Web Services and REST APIs

Hans-Petter Halvorsen

Table of Contents

Problem

How to Share Data between Devices in a Network?



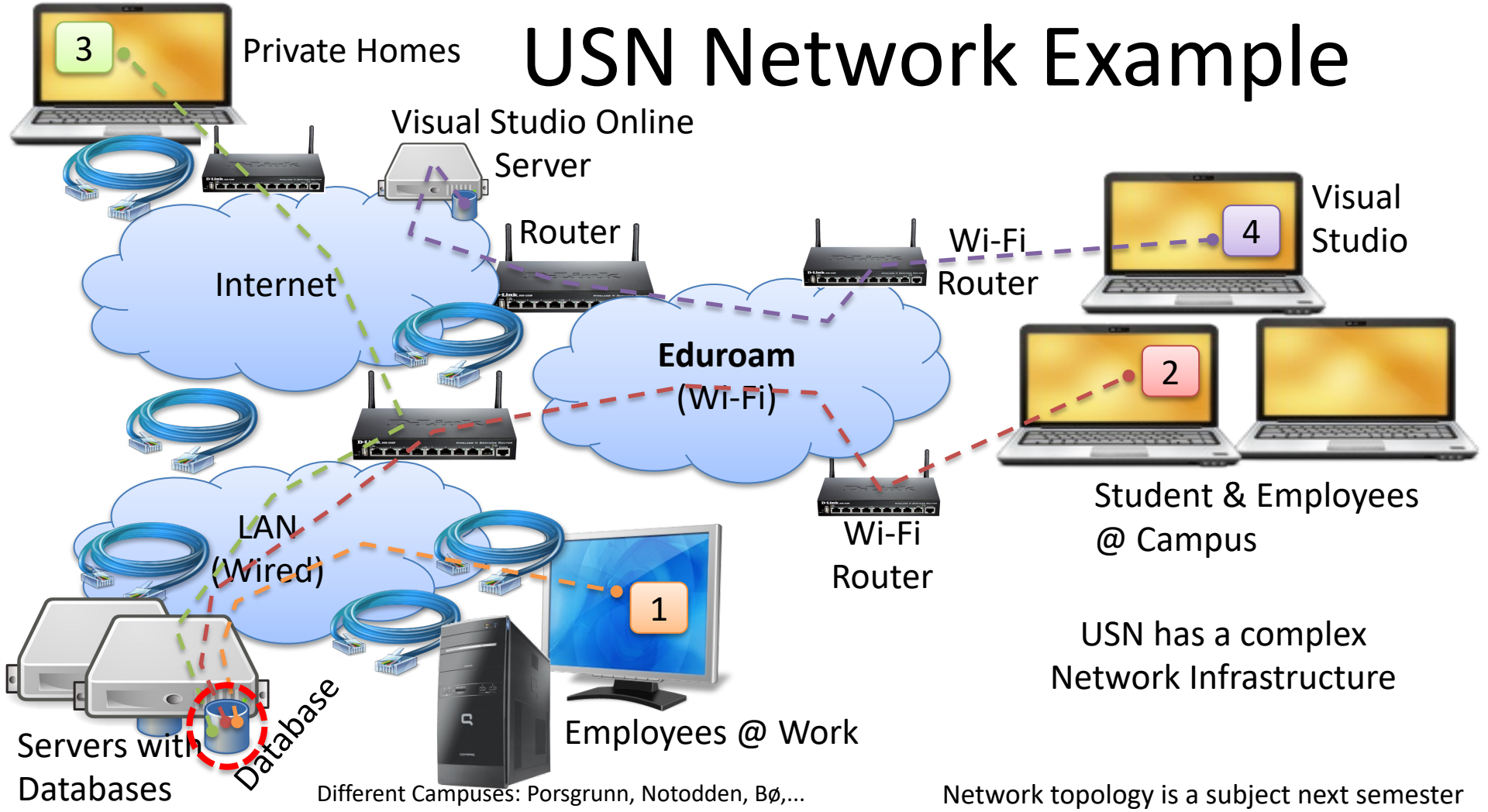
Problem

How to Share Data between Devices in a Network?

Direct Connection between the Database and the Clients that need the Data is normally not possible, due to security, compatibility issues, etc. (Firewalls, Hacker Attacks, etc.)

Direct Connection in a Local Network (behind the Firewall) is normally OK – but not over the Internet!!

USN Network Example

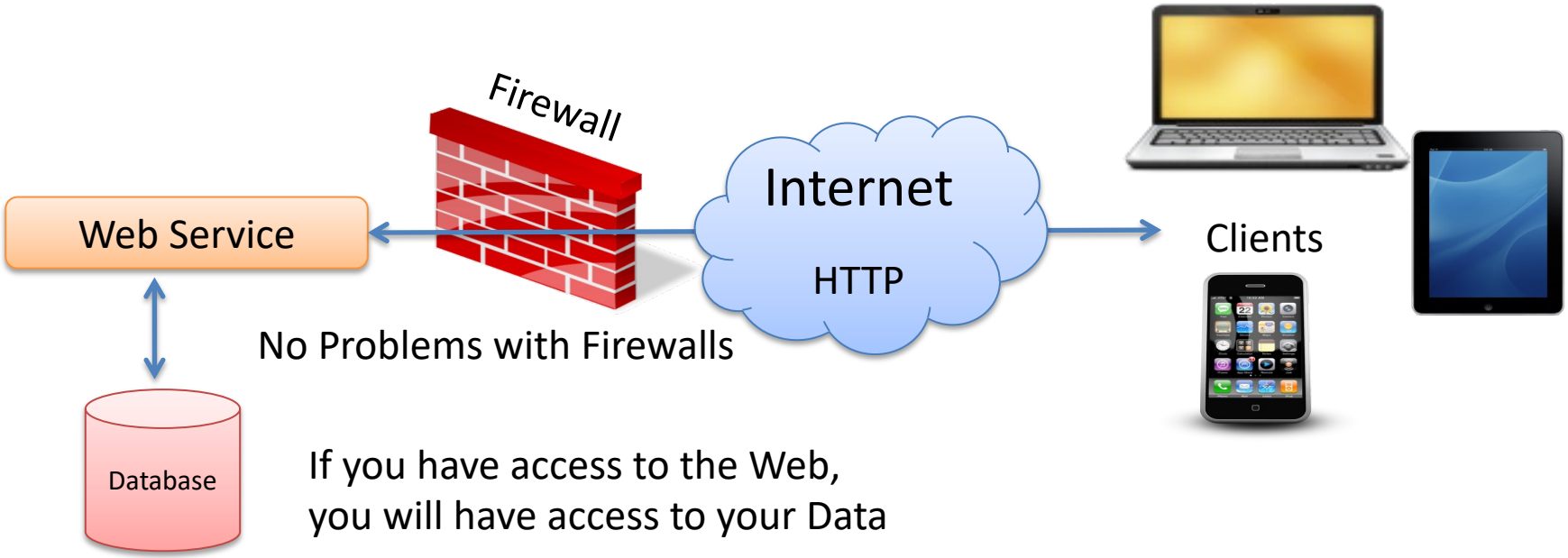


USN has a complex
Network Infrastructure

Different Campuses: Porsgrunn, Notodden, Bø,...

Network topology is a subject next semester

Solution: Web Service



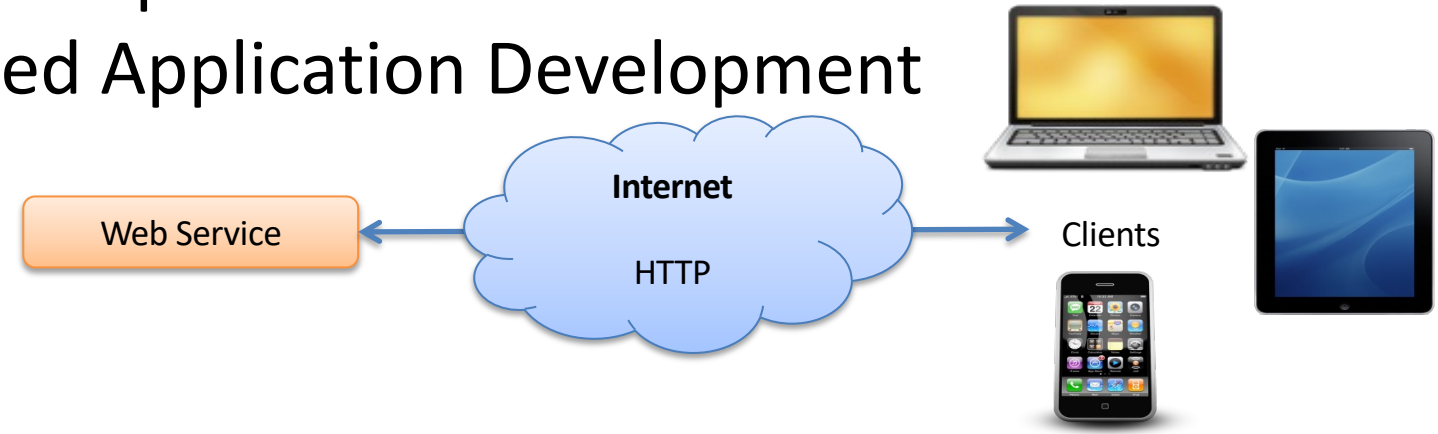
Web Services uses standard Web Protocols like HTTP, etc.
HTTP is supported by all Web Browser, Servers and many Programming Languages

Web Services

- A Web service is a method of communications between two devices over the World Wide Web.
- Web API
- Standard defined by W3C
- Cross-platform
- Web Services can be implemented and used in most Programming Languages (C#/ASP.NET, PHP, LabVIEW, Objective-C, Java, ...)
- Uses standard Web Technology and Web Protocols
  - HTTP, REST, SOAP, XML, WSDL, JSON, ...

Why Web Service?

- Today Web Services have been very popular
- Easy Data sharing over Internet
- Platform-independent Communication
- Makes it possible of integration of different systems and platforms
- Distributed Application Development

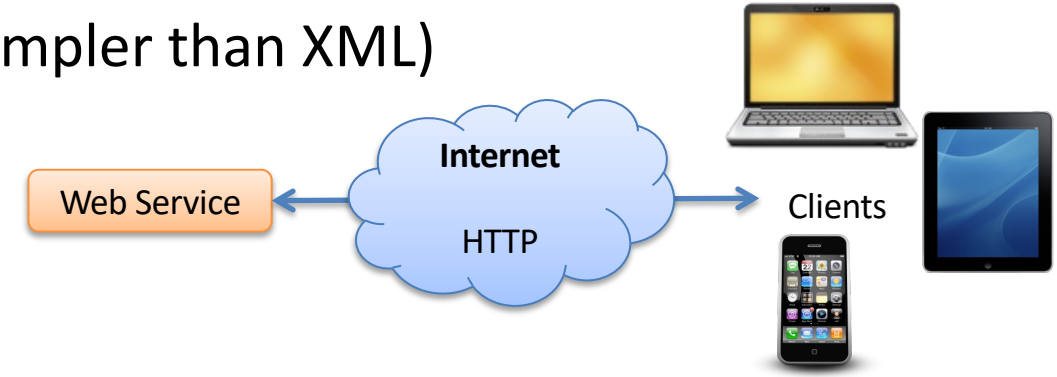


Web Services

- Web Services 1.0: Uses SOAP
  - “Complex”, XML based

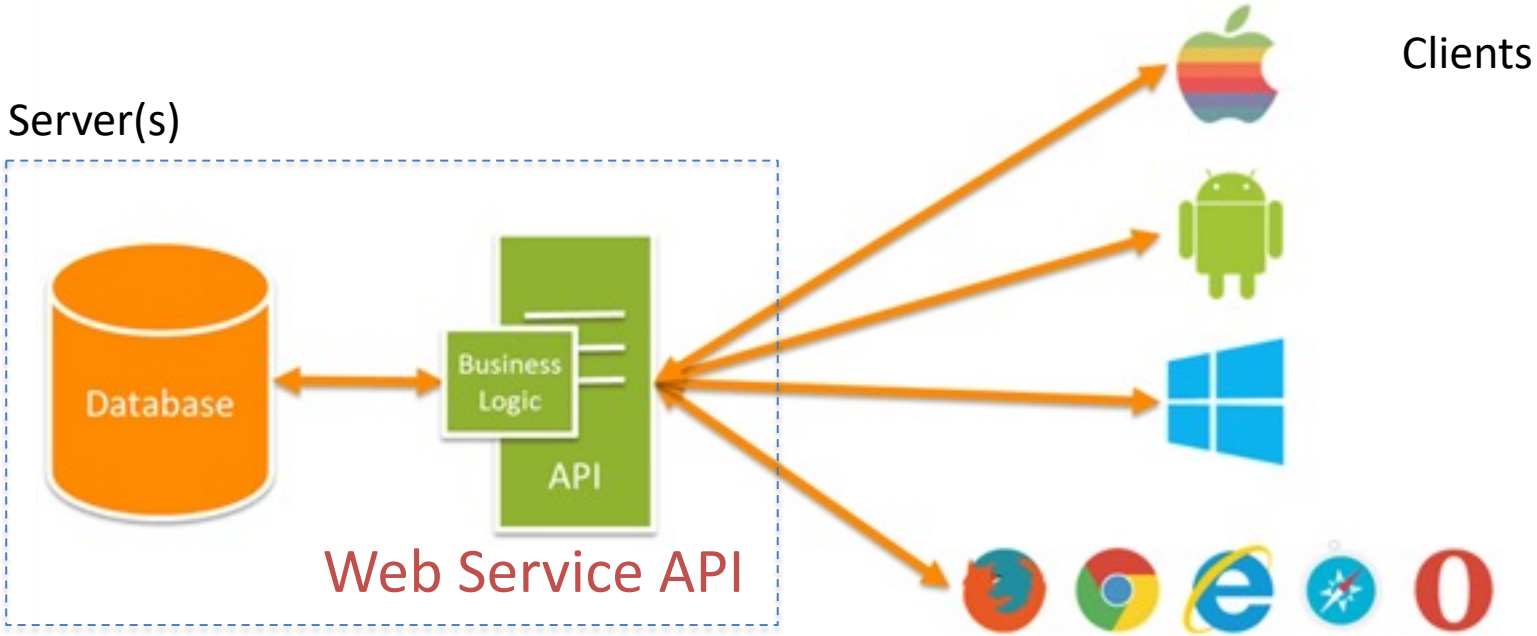
Visual Studio: ASP.NET ASMX Web Service

- Web Services 2.0: Uses REST (Referred to as REST API)
  - Lightweight and Flexible, Less Complex than using SOAP, The preferred model today
  - JSON or XML (JSON is simpler than XML)



Visual Studio: ASP.NET Web API

Web Service Architecture Example



Each app uses the same API to get, update and manipulate data. All apps have feature parity and when you need to make a change you just make it in one place in line with the 'Don't Repeat Yourself' (DRY) principle of software development. The apps themselves then become relatively lightweight UI layers.

http://blogs.msdn.com/b/martinkearn/archive/2015/01/05/introduction-to-rest-and-net-web-api.aspx

https://www.halvorsen.blog

Web Architecture

Hans-Petter Halvorsen

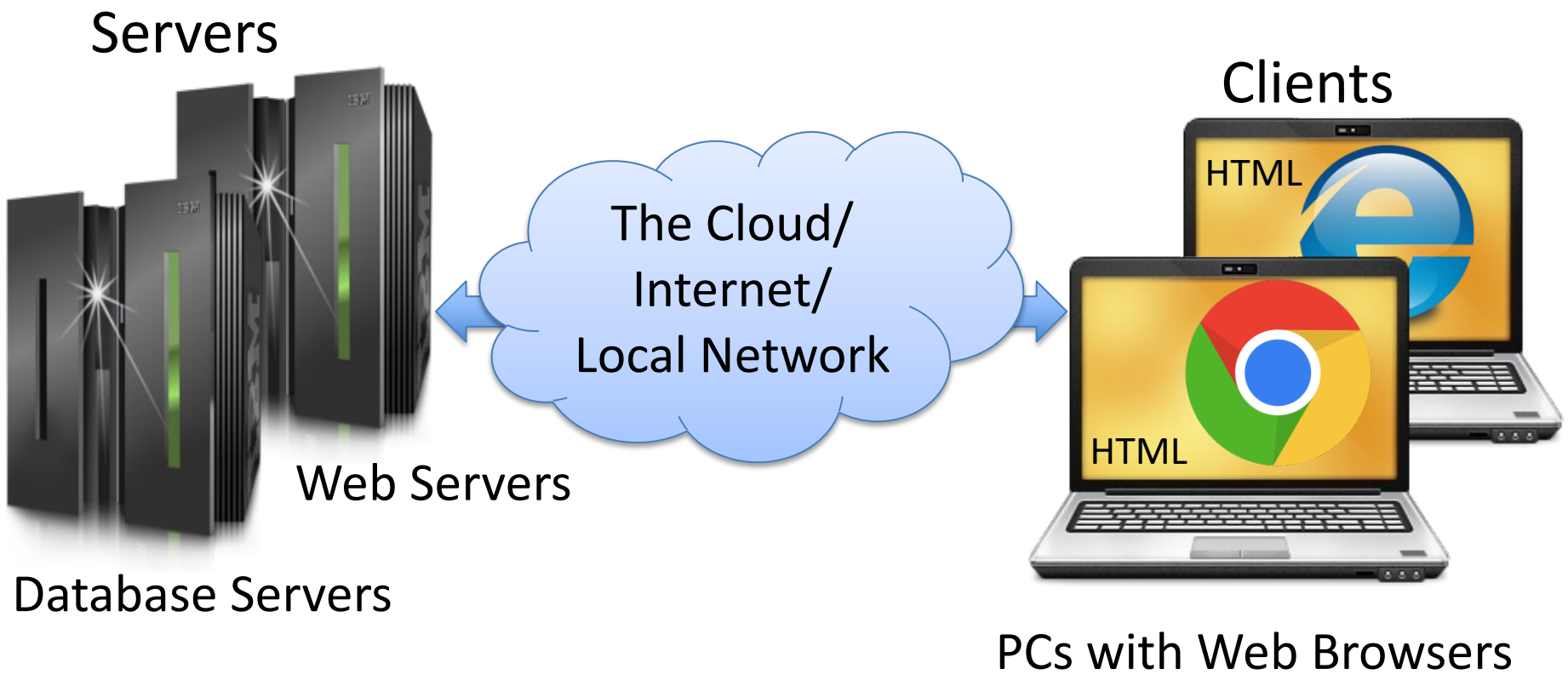
Table of Contents

Web Architecture

The Web uses all these architecture principles mentioned

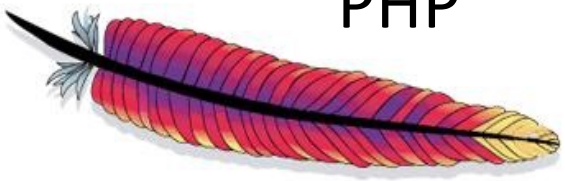
- Client-Server
- 3-tier/n-tier
- MVC
- APIs
- Web Services/REST APIs
- etc.

Web Architecture



Web Server Platforms

(pronounced "engine x")
- Has become very popular lately



PHP

Apache

Internet Information Services

ASP.NET

Cross-platform: UNIX, Linux, OS X, Windows, ...

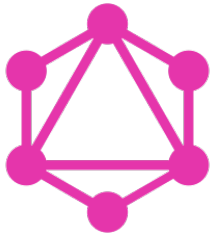
The term web server can refer to either the hardware (the computer) or the software (the computer application) that helps to deliver web content that can be accessed through the Internet.
The most common use of web servers is to host websites, but there are other uses such as gaming, data storage or running enterprise applications.

REST API/Web API

- REST - Representational State Transfer
- Uses HTTP as Communication Protocol
- GET, PUT, POST, DELETE
- Use the JSON Data Format
  - customer = { "name": "John", "address": "Highway 37" }

https://en.wikipedia.org/wiki/Web_API

GraphQL



- GraphQL is a query language for APIs
- It provides an efficient, powerful and flexible approach to developing Web APIs, and has been compared and contrasted with REST and other web service architectures
- GraphQL is a new concept, originally made by Facebook in 2012
- GraphQL is now Open Source
- https://graphql.org

Implementing

Hans-Petter Halvorsen

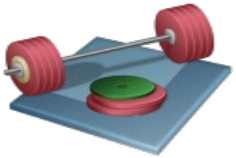
Table of Contents

Architecture and Implementation

Tasks:

- Cont. Implementing your System according to the SRD
- Use Azure DevOps
- Make sure <u>your</u> Architecture is documented (with different sketches and written information) in the Software Requirements and Design (SRD) document
- Use the terms learned in this topic

Software Architecture & Deployment



Teknisk arkitektur, installasjon og utrulling av programvare til kunden:

- Alle applikasjoner må installeres, enten det er en desktop-applikasjon, mobil-applikasjon eller en webside eller en web-applikasjon.
- Systemets arkitektur gjenspeiles i hvordan systemet kan «deployes»/installeres.
- Det er viktig at man tidlig har en formening hvordan dette skal gjøres for deres produkt/system.

Software Architecture & Deployment

Teknisk arkitektur, installasjon og utrulling av programvare til kunden:

- F.eks. installerer man Outlook lokalt på maskinen sin ved å trykke på en setup.exe fil for deretter å gå gjennom en såkalt trinnvis "installasjons-wizard" hvor brukeren må gjøre ulike valg underveis.
- Denne setupen installerer typisk selve outlook applikasjonen men også noen services, lokalt repository, m.m.
- Server biten av Outlook, dvs Exchange må også installeres (på en Server) slik at Outlook klientene kan koble seg på denne.
- Exchange består typisk av en database, diverse services, APIer, websider, m.m. som må installeres.

Software Architecture & Deployment

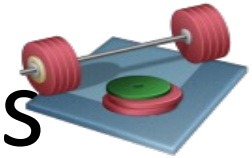
Stikkord:

- Gi oversikt over den tekniske arkitekturen til produktet deres
- Forklar og illustrer (tekst og skisser) hvordan deres løsning skal installeres og tas i bruk av kunden.
- Gjerne beskriv og illustrer ulike muligheter (kunden ønsker fleksibilitet).
- Hva slags dokumentasjon trenger dere å utarbeide ifm installasjon, drift og vedlikehold?
- Forutsetninger og krav?
- Krav til hardware og software? Hva slags programvare trengs å installeres før kunden kan installere produktet som dere lager? m.m.
- Krav til servere? Krav til klienter? Hardware og software ifm dette
- Database? Arkitektur og installasjon ifm denne?
- Sikkerhet og tilgang?
- osv.

Oppdatere SRD dokumentet med denne informasjonen

Bruk alle tilgjengelige resurser for å finne svar på disse tingene, internett, lærebøker, kompendium, fagets websider, m.m.

Deployment Architecture Scenarios



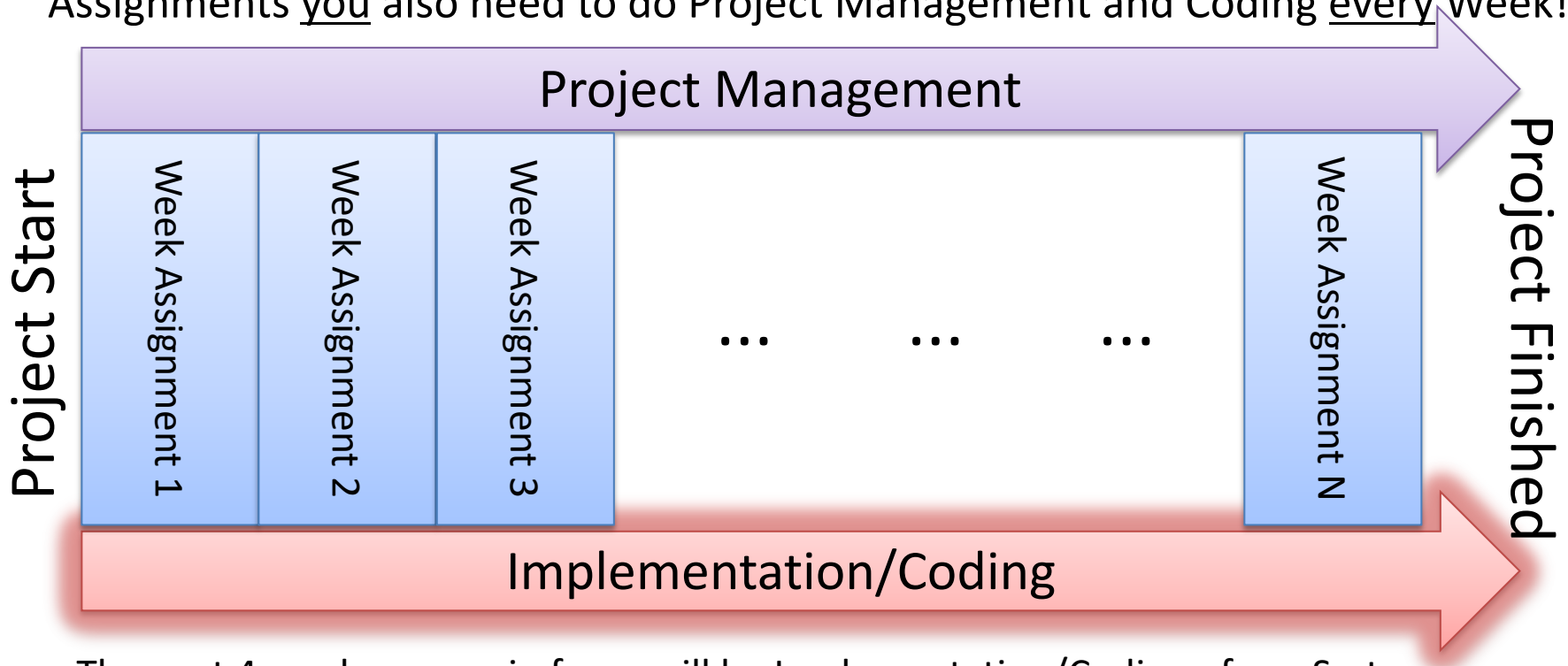
Forskjellige deployment-scenarier kan være:

- Alternativ 1: Kunden ønsker å installere og drifte løsninger deres i sitt eget bedriftsnettverk. Kunden må kjøpe inn en ny server/flere servere som de kan bruke til dette formålet – eller bruke en eksisterende server de har.
- Alternativ 2: Kunden har ikke egen infrastruktur og serverpark, kunden har heller ikke dypgående kunnskap om drift og vedlikehold av slike systemer.

- Systemet deres bør støtte begge disse scenariene.

This Course/Project

In Class we have focus on the Week Assignments – But In addition to the Week Assignments you also need to do Project Management and Coding every Week!



The next 4 weeks our main focus will be Implementation/Coding of our System

https://www.halvorsen.blog

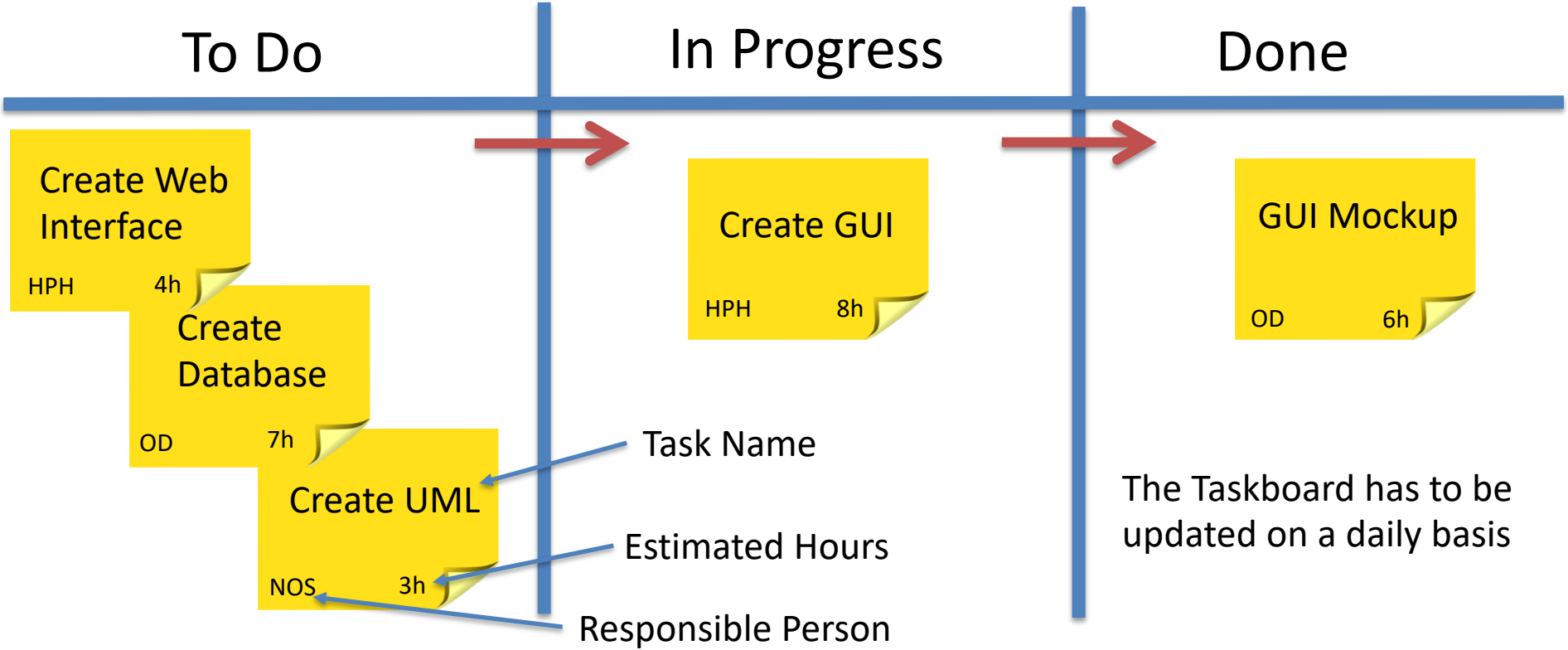
Scrum and Scrum Meetings

Hans-Petter Halvorsen

Table of Contents

Sprint Task Board

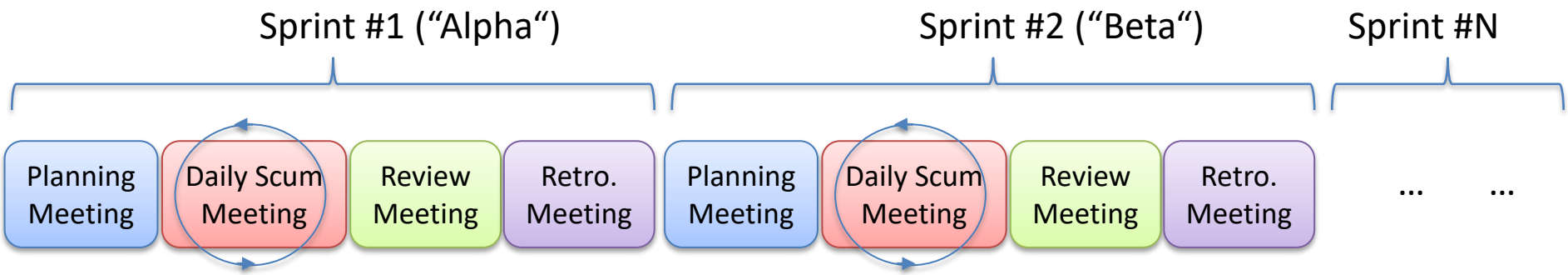
Make sure to use and update the Taskboard in Azure DevOps on a daily basis!!



Scrum Meetings

- Sprint Planning Meeting
  - The purpose with the Sprint Planning is to discuss and select the work items for the next Sprint.
  - You select work items from the **Product Backlog** into the next **Sprint Backlog**.
- Daily Scrum Meeting (Standup Meeting). 3 Questions:
  1. What has been accomplished since last meeting?
  2. What will be done before the next meeting?
  3. What obstacles are in the way?
- Sprint Review Meeting
  - The purpose with the Sprint Review is to have a complete review of all the tasks/features that should be completed in the Sprint (Sprint Backlog items)
  - In this meeting, your team **demonstrates the features** that it completed in the sprint
- Sprint Retrospective Meeting
  - Learn form previous Sprints. Find Improvements the Scrum Team will agree on for the next Sprint.
  - Making Actionable Commitments
    1. Keep Doing
    2. Start Doing
    3. Stop Doing

Scrum Meetings



The Scrum Master is responsible for that these meetings are held

Daily Scrum Meeting

3 Questions:

1. What did you do yesterday (since last meeting)?
2. What shall you do today (until the next meeting)?
3. Any Problems?

Purpose with the Daily Scrum Meeting:
- Synchronize activities and create a plan for next 24 hours.
- Track Progress

References

- SOAP vs. REST Challenges: http://www.soapui.org/The-World-Of-API-Testing/soap-vs-rest-challenges.html
- Web Service: http://en.wikipedia.org/wiki/Web_service
- ASP.NET Web API Example: http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api
- Introduction to REST and .NET Web API: http://blogs.msdn.com/b/martinkearn/archive/2015/01/05/introduction-to-rest-and-net-web-api.aspx

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog