Desarrollo de Aplicaciones Web Empresariales

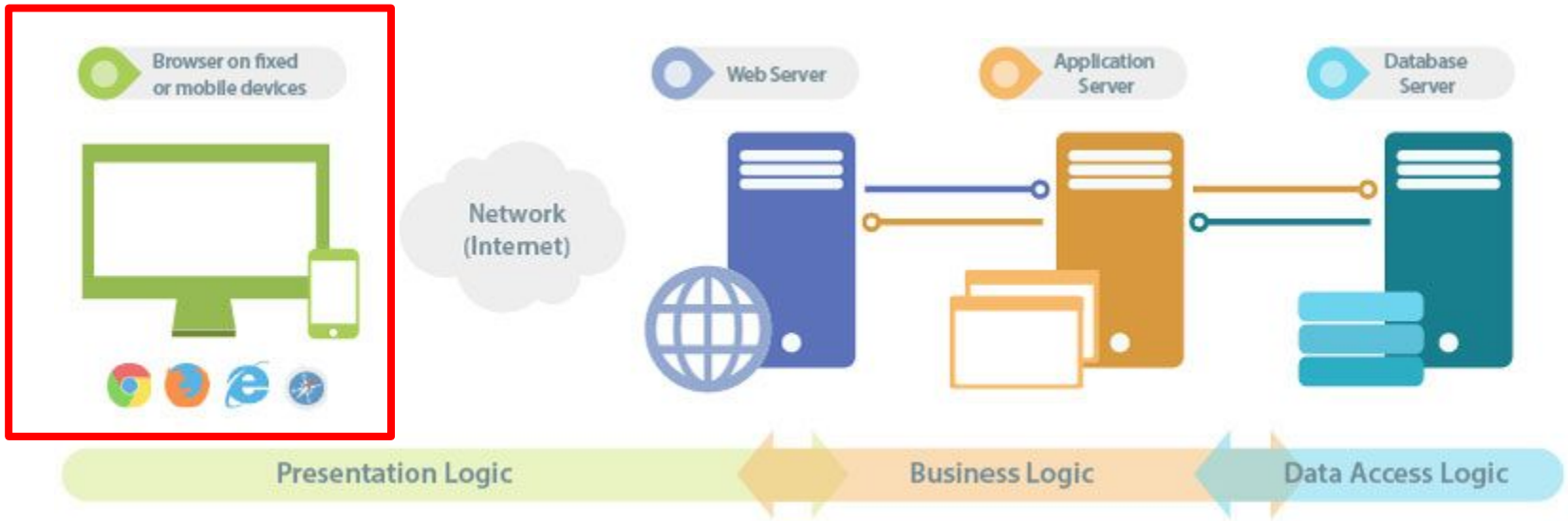
Exequiel Fuentes Lettura
exequiel.fuentes@ucn.cl

Información de contacto

- Exequiel Fuentes Lettura
  - Email: exequiel.fuentes@ucn.cl
  - Horario de Atención: Jueves y Viernes, bloque C

- Departamento de Ingeniería de Sistemas y Computación
  - Oficina: Y1 - 329
  - http://www.disc.ucn.cl

Arquitectura de aplicaciones Web



Breve historia de aplicaciones Web

- Al principio solo eran páginas estáticas.
- Se utilizaba Common Gateway Interface (CGI):
  - La URL se mapeaba para encontrar encontrar el programa para generar la página.
  - El programa terminaba cuando la página se enviaba.
  - Cada solicitud al servidor no tenía estado y eran independientes.
- Perl dominaba el escenario.

Primera generación

- Ejemplos: PHP, ASP.net, Java servlets
- Incorpora lenguajes en el lado del servidor.
- Plantillas: mezcla de código con HTML.
- Las librerías contemplaban:
  - Manejo de URL.
  - Generación de HTML.
  - Sesiones.
  - Interfaces para base de datos.

Segunda generación

- Ejemplos: Ruby on Rails, Django.
- Se descompone la aplicación en Modelo, Vista y Controlador.
- Introduce Object-relational mapping (ORM) para simplificar la utilización de base de datos: Tablas son clases

Tercera generación

- Ejemplos: AngularJS, ReactJS, etc.
- Se cuenta con un framework de Javascript que se ejecuta en el browser.
- Dirigido a aplicaciones interactivas.
- El framework no depende de las capacidades del servidor:
  - Servidor usando Javascript: Node.js
  - Base de datos No-SQL (ejemplo: MongoDB)
- Muchos conceptos de la generación anterior se tomaron:
  - Modelo, Vista y Controlador.
  - Plantillas.

Model-View-Controller (MVC)

- Model: Maneja los datos de la aplicación.
  - Objetos Javascripts.
- View: Es como luce la página.
  - HTML/CSS.
- Controller: Maneja el modelo, las vistas y las interacciones que solicita el usuario.
  - Código en Javascript.

Generación de la Vista

- Web app: Necesita generar HTML y CSS.
- Las plantillas son las técnicas más utilizadas. La idea básica es:
  - Escribir HTML que contiene partes de la página que será siempre la misma.
  - Agregar el código que generará las partes que son dinámicas de cada página.
- Beneficios de las plantillas:
  - Fácil para visualizar la estructura del HTML.
  - Fácil para ver como el código dinámico se complementa.
  - Puede hacerse el servidor o en el browser.

Plantilla en AngularJS

- AngularJS es un lenguaje rico en plantillas, pero se verá luego.

```
<body>
    <div class="greetings">
        Hello {{models.user.firstName}},
    </div>
    <div class="photocounts">
        You have {{models.photos.count}} photos to review.
    </div>
</body>
```

Controladores

- Tercera generación: Javascript ejecutándose en el browser.
- Responsabilidades:
  - Conectar el modelo y las vistas. Comunicación con el servidor.
  - Controlar las plantillas.
  - Manejar las solicitudes del usuario.

Controlador en AngularJS

- AngularJS crea $scope y llama a la función controladora cuando la vista es instanciada.

```
function userGreetingView ($scope, $modelService) {
    $scope.models = {};
    $scope.models.users = $modelService.fetch("users");
    $scope.models.photos = $modelService.fetch("photos");
    $scope.okPushed = function okPushed() {
        // Code for ok button pushing
    }
}
```

Modelo de datos

- Es toda la información no estática que requiere las vistas o los controladores.
- Tradicionalmente se refiere a una base de datos. Se usa ORM.
- El esquema de base de datos no cambia frecuentemente, esto depende del modelo de datos de la aplicación o nuevos requerimientos.

AngularJS no tiene un modelo de datos

- AngularJS proporciona soporte para obtener datos desde el servidor Web:
  - REST.
  - JSON.
- En el servidor si se utiliza MongoDB se ve algo así:

```
var userSchema = new Schema({
    firstName: String,
    lastName: String,
});
var User = mongoose.model('User', userSchema);
```

Model-View-Controller

- Es utilizado ampliamente por todos los frameworks modernos.
- Cómo determinar las partes:
  - Cuales son los componentes de la aplicación?
  - Cómo lucen los componentes de la aplicación?
  - Qué datos son requeridos?
  - Cuales funcionalidades se necesitan controlar?
- Aquí también hay problemas:
  - Diseño modular.
  - Componentes reusables.
  - Pruebas.
  - Etc.

¿Preguntas?