Conceptualización Técnica

Proyecto de Ingeniería de Software.

Grupo 10 - The New Persians

Facultad de Ingeniería, UDeLaR

[**MVC.**](#_heading=h.va9m1tyy4mzv) **5**

[**Cliente / Servidor.**](#_heading=h.6rr5151czql) **6**

[**Diagrama de contexto del sistema.**](#_heading=h.xerxkitb0spc) **9**

[**Diagrama físico**](#_heading=h.xlg2d3vttvls) **10**

[**Diagrama de Componentes**](#_heading=h.s9n6aucu234y) **11**

[**Modelo Entidad-Relación**](#_heading=h.woh2iavqd9fq) **12**

Introducción.

Este documento tiene como objetivo generar una conceptualización técnica que determine cuál será la base de la arquitectura del proyecto.

Tecnologías a utilizar.

A continuación se describen de forma breve aquellas tecnologías que el equipo utilizará para la construcción del producto propuesto por el cliente. A modo de comentario, se deja constancia de que todas estas tecnologías fueron propuestas por el cliente y no por el equipo de desarrollo.

1. Frontend
   * Javascript: JavaScript es un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [interpretado](https://es.wikipedia.org/wiki/Int%C3%A9rprete_(inform%C3%A1tica)), que sigue el estándar [ECMAScript](https://es.wikipedia.org/wiki/ECMAScript). Se define como [orientado a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos), [basado en prototipos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_basada_en_prototipos), imperativo, [débilmente tipado](https://es.wikipedia.org/wiki/Tipado_fuerte#Lenguajes_no_tipados) y [dinámico](https://es.wikipedia.org/wiki/Tipado_din%C3%A1mico). JavaScript es el único lenguaje de programación que funciona en los navegadores de forma nativa y se utiliza principalmente en el desarrollo de aplicaciones web del [lado del cliente](https://es.wikipedia.org/wiki/Cliente_(inform%C3%A1tica)).
     1. React: React es una biblioteca [Javascript](https://es.wikipedia.org/wiki/Javascript) de [código abierto](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto). Es mantenido por [Facebook](https://es.wikipedia.org/wiki/Facebook) y la comunidad de [software libre](https://es.wikipedia.org/wiki/Software_libre). React intenta ayudar a los desarrolladores a construir [aplicaciones](https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_inform%C3%A1tica) que usan datos que cambian todo el [tiempo](https://es.wikipedia.org/wiki/Tiempo). Su objetivo es ser sencillo, declarativo y fácil de combinar. React será la vista en el contexto del patrón [MVC](https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador) (Modelo-Vista-Controlador).
   * HTML: HTML es el lenguaje de marcado que se usa para estructurar y dar significado semántico a un sitio web, por ejemplo, definiendo párrafos, encabezados y tablas de datos, o insertando imágenes y videos en las diferentes páginas.
   * CSS: CSS es un lenguaje de reglas de estilo que se utiliza para aplicar estilo al contenido HTML, con CSS se pueden establecer colores de fondo, cambiar el tipo de fuente, entre otras cosas.
2. BACKEND
   * PostrgreSQL: PostgreSQL es un [sistema de gestión de bases de datos](https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos) [relacional](https://es.wikipedia.org/wiki/Modelo_relacional) de [código abierto](https://es.wikipedia.org/wiki/Software_de_c%C3%B3digo_abierto), como muchos otros proyectos de [código abierto](https://es.wikipedia.org/wiki/Software_de_c%C3%B3digo_abierto), el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada. PostgreSQL no tiene un gestor de defectos, haciendo muy difícil conocer el estado de sus defectos.
   * Ruby on Rails
     1. Ruby: Ruby es un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [orientado a objeto](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos)s y su sintaxis se inspira en [Python](https://es.wikipedia.org/wiki/Python) y [Pe](https://es.wikipedia.org/wiki/Perl)rl. Es un lenguaje interpretado en una sola pasada y su implementación oficial es distribuida bajo una [licencia](https://es.wikipedia.org/wiki/Ruby#Licencia) de [software libre](https://es.wikipedia.org/wiki/Software_libre).
     2. Rails: Rails, es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby el cual se utiliza para desarrollar aplicaciones que utilicen el patrón Modelo Vista Controlador para su arquitectura.
     3. Active Record: Active Record es el ORM que utiliza ruby on rails para mapear estructuras de una base de datos relacional, sobre una estructura lógica de entidades.

Patrones Arquitectónicos.

A continuación se listan y describen los patrones arquitectónicos que se considerarán a ser usados en el diseño de la arquitectura del sistema.

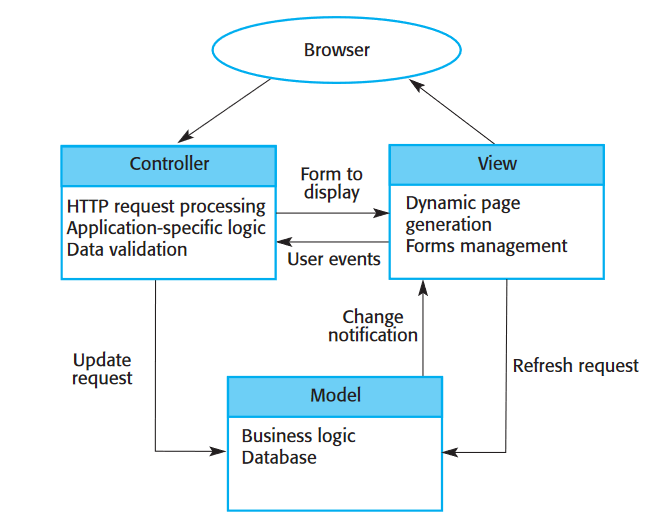
## MVC.

El patrón arquitectónico MVC fue el primero en evidenciarse, dado que la tecnología que utilizamos para este proyecto (RoR) es un framework que se define bajo este paradigma.

En términos generales el Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software que separa la lógica de una aplicación, sus datos, su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador; es decir, por un lado define componentes para la representación de la información y por otro lado para la interacción con el usuario.

MVC bajo nuestra realidad es el clásico ejemplo de una aplicación WebApp que utiliza este paradigma, más precisamente:

* El Modelo: Maneja todas todo los datos del sistema referido a las Encuestas, Usuarios, Grupos, etc. (RoR)
* El Controlador: Maneja todos los eventos generados a partir de la interacción del usuario con la WebApp. (RoR)
* La Vista: Es el encargado de proveer las vistas y cómo los datos son presentados al usuario. (React)



## Cliente / Servidor.

El modelo anterior sirve para generar un contexto sobre el manejo interno de la aplicación pero no hay que obviar el hecho de que nuestra WebApp se integra al cliente como un servicio externo, servicio que será utilizado por NeoCoast y sus clientes a través de un navegador. Por eso es necesario considerar el patrón arquitectónico Cliente / Servidor.

En una arquitectura Cliente / Servidor el sistema es presentado como una colección de servicios, con cada servicio siendo potencialmente asociado con un servidor. Los clientes son usuarios de estos servicios y acceden al/a los servidor/es para utilizarlos.

Nuestro sistema, en rasgos generales es presentado al cliente como una colección de servicios de Encuestas que diferentes tipos de usuarios utilizan (por ejemplo, el Administrador crea Encuestas y les programa una agenda, mientras que los empleados o invitados las responden) , que están alojados en la infraestructura provista (Heroku Server).

Se trata entonces de una aplicación multi-cliente que consume un determinado conjunto de servicios concurrentemente. Es necesario estimar la carga de trabajo de cada uno de estos servicios así como la periodicidad para determinar la cantidad de servidores a montar:

* Cuánto es la carga de usuarios.
* Cuánto cuesta, en tiempo de cómputo, analizar el histórico de las encuestas y el filtrado por parámetros.
* Analizar escenarios de stress (Todos los empleados de NeoCoast y clientes de los mismos respondiendo encuestas el mismo día en la misma ventana de tiempo).

Diagramas

A continuación se presentan los diagramas que modelan diferentes aspectos de la conceptualización técnica. Nota: Algunos se presentan rotados para que se puedan apreciar mejor.

## 

## Diagrama de contexto del sistema.

## Diagrama físico

## Diagrama de Componentes

## Modelo Entidad-Relación