

INSTITUTO SUPERIOR DE FORMACIÓN TÉCNICA N° 190

CARRERA: TÉCNICO SUPERIOR EN ANÁLISIS DE SISTEMAS

ESPACIO CURRICULAR: PRÁCTICAS PROFESIONALIZANTES

CURSO: TERCER AÑO

ALUMNO/S:

- Cardozo José Ignacio
- Castelli Juan Ignacio
- Larracoechea Ignacio
- Rossi Gino
- Sardinez Gerardo
- Telayna Iñaki

PROFESOR: Daniel Benvenuto

CICLO LECTIVO: 2020

ÍNDICE

ÍNDICE	1
PROYECTO ISFT N°190	3
Introducción	3
Objetivos	4
Etapas del proyecto	4
Requerimientos	7
Reqfun1--Desplegar info INST	7
ReqNOfun2--Buscador	8
Reqfun3-Inscripción	8
Reqf3a--Preinscripción	9
Reqf3b--Generación de Usuario	9
Reqfun4--Noticias	10
Reqfun5--Login (inicio de sesión)	10
Reqfun5a--Recuperación de contraseña(usuario)	11
Reqfun6--Gestión de Trámites	11
ReqNOfun7--Diseño Responsive	12
Diseño del sistema	13
Elección de tecnologías:	13
¿Por qué Node.js?	13
¿Por qué Express?	13
¿Por qué Postgresql ?	13
¿Por qué Sequelize?	14
Patrón de diseño MVC	15
GIT	15
Configuraciones iniciales	16
Instalación y configuración de Express	16
Configuración de PostgreSQL	18
Configuración de Git	19
Configuraciones iniciales	19
Inicialización de un nuevo Repositorio Local	19
Comandos básicos:	19
Instalación y configuración de sequelize	27
Generación de modelos	29
Generación de migración	30
Sembrado de modelos	32
Mackaroo	33

Instalación y configuración de Passport	34
Implementación del proyecto	37
WEB ACTUAL ISFT 190	46
Casos de Uso	47
Diagramas	48
Página Principal	60
Frontend	60
BackEnd	61
BackEnd funcionalidades implementadas	62
Desplegar Información del Instituto	62
Login de usuario a nivel Administrativo	62
Login de usuarios a nivel Alumno	63
Conclusión	64
Glosario	65
Diagramas	66
Bibliografía	67

PROYECTO ISFT N°190

Introducción

Estando cursando la materia Prácticas Profesionalizantes nos vimos en la necesidad de idear un trabajo final cuyo objetivo es la integración de los conocimientos adquiridos durante el transcurso de la carrera.

En base a dicho requerimiento, optamos por implementar como proyecto de trabajo la renovación de la página web del instituto. La misma se encuentra diseñada en una versión obsoleta de un gestor de contenidos (joomla), lo cual dificulta el mantenimiento del sistema y la escalabilidad del mismo.

A raíz de esto, se emplearon tecnologías modernas mucho más flexibles, con documentación abundante, la cual facilita la curva de aprendizaje de las mismas permitiendo el posterior mantenimiento, desarrollo y escalabilidad del sistema.

A lo largo del informe se podrá apreciar el análisis, diseño, desarrollo e implementación de una serie de optimizaciones, las cuales facilitan tanto el trabajo por parte del personal administrativo del instituto, como también una mejor experiencia de usuarios.

Entre las optimizaciones tenidas en cuenta se pueden mencionar :

- Diseño moderno
- Responsive (adaptabilidad a distintos dispositivos)
- Gestor de usuarios
- Gestión de trámites automatizados
- Administración de la página (CRUD de noticias y carreras, etc.)

Objetivos

- **Optimización y reestructuración de la página del Instituto.**

Debido a que se precisa de una página institucional con la cual interactuar con la comunidad educativa, se busca la modernización tanto visual como funcional de la misma. Siendo de esta manera intuitiva y que pueda desplegar información de interés no solo a los alumnos sino también a aquellos interesados en conocer las actividades del instituto.

- **Reestructuración del Sistema de login de la página.**

El sistema actual si bien contaba con un sistema de login, el mismo no brindaba una funcionalidad. Por tal motivo, se rediseñará el mismo implementando una jerarquía de usuarios los cuales dispondrán de distintos privilegios y funcionalidades al momento de interactuar con el sitio.

- **Sistema de gestión de alumnos.**

Se adicionará un sistema de trámites automatizados, el cual reduce la carga de trabajo al personal administrativo, agilizando la gestión de los mismos sin necesidad de concurrir a la institución. Entre los trámites habilitados se encuentran:

- Solicitud de Alumno regular.
- Solicitud Analítico.
- Solicitud Inscripción a exámenes.
- Solicitud Inscripción a materias.
- Solicitud Historial de carreras.

Etapas del proyecto

1. Analizar el sistema actual

Se tomará todo el tiempo necesario para analizar la página actual del Instituto, para sacar referencias y ver que mejoras se le deben hacer, para el futuro rediseño del mismo.

2. Prototipado para demostración al contratante.

Se realizará un prototipado de la página rediseñada para mostrar al instituto. Para ver si las mejoras producidas son aceptables, y ver qué componentes adicionales se le podrían implementar.

3. Desarrollo y documentación del software.

1. Selección de las tecnologías a implementar, tanto en el Front-End como en el Back-End. También se elige el área de trabajo para desarrollar dichas tecnologías.

2. Diseño del software, teniendo en cuenta los siguientes recursos del sistema a diseñar:
 - Diagrama de Contexto
 - Diagrama/s de Flujo de Datos
 - Diagrama Entidad-Relación
3. Prueba del software diseñado, para realizar las pruebas que sean necesarias. Todo con la intención de validar y verificar la eficacia del mismo.
4. Documentar el desarrollo del software, tecnologías que se implementaron y los problemas que pudo haber con las mismas.

4. Prueba para el nuevo sistema.

Se seleccionarán a los usuarios que se necesite para probar la eficacia del sistema. Que en este caso serán: Usuario Invitado, un Alumno y uno que haga rol de Administrador.

5. Desarrollar los requerimientos para la nueva Página:

1. Desplegar información sobre el INST(Front)
 - a. Dirección
 - b. Carreras
 - c. Contacto
 - d. Información inscripción - preinscripción.
 - e. Posicionamiento SEO
2. Buscador(Back-Front)
3. Inscripción
 - a. Alta de ingresante (requiere una validación por parte del instituto)

Requisitos necesarios para inscripción

- Realizar la preinscripción en el siguiente link: [click aqui](#)

Luego dirigirse personalmente al Instituto a fin de confirmar la inscripción, llevando además:

* Formulario de Preinscripción online IMPRESO.

* 1 (una) fotocopia de D.N.I.

* 1 (una) fotocopia Título Secundario (Nacionalizado) o Constancia de Título en Trámite.

* 1 (una) foto Carnet 4x4

* Certificado de Buena Salud (otorgado por médico de cabecera o centro de salud)

* Abonar el Arancel de Cooperadora (Anual)

4. Noticias
 - a. Tratamiento de imágenes(subida al servidor)
 - b. Categorías

- c. Compartir en redes sociales
- 5. Login usuario (CRUD)
 - a. Recuperar contraseña
 - b. Información disponibles para los usuarios
 - i. Jerarquia de usuarios
 - ii. Invitado
 - iii. Alumno
 - iv. Administrativo
 - c. Limitar la cantidad de intentos de login fallidos (Backend)
 - d. Restricción de acceso a nivel vista(frontend) y a nivel datos(backend)
- 6. Gestión de Trámites
 - a. Solicitud Certificado alumno regular / historial carrera / analitico / etc. Y generación de comprobante
 - b. Solicitud Beca Progresar
 - c. Cualquiera de los trámites anteriores sea automático a través de una plantilla en la que se complete con nombre, apellido y documento del alumno y que la firma ya esté digitalizada.
 - d. Respuesta a consulta (ej, info sobre carreras)(Formulario)
- 7. Responsive

Requerimientos

Reqfun1--Desplegar info INST

Identificador	<i>ReqFun1</i>
Categoría	<i>Funcional</i>
Descripción Corta	<i>Desplegar información sobre el Instituto.</i>
Descripción Detallada	<i>El sistema deberá proporcionar al usuario de manera intuitiva información respecto a la ubicación geográfica del instituto, así como las carreras que se dictan, lo cual incluye no solamente el nombre de dichas carreras, sino que también se deberá detallar información relevante respecto a cada una (Incumbencias ,alcance, etc.) A su vez también se tendrá que proporcionar información de contacto, asimismo cómo información relevante para los nuevos alumnos (aspirantes/ingresantes)</i>
PreCondiciones	<i>El usuario deberá estar situado en la página de inicio.</i>
Dependencia	
Actor	<i>Usuario (Invitado/Alumno/Administrador)</i>

ReqNOfun2--Buscador

Identificador	<i>ReqNOFun2</i>
Categoría	No Funcional
Descripción Corta	<i>Buscador</i>
Descripción Detallada	<i>El sistema debe brindar una opción al usuario de solicitar una búsqueda por la página.</i>
PreCondiciones	<i>El usuario debe estar parado en el buscador de la página.</i>
Dependencia	
Actor	<i>Usuario (Invitado/Alumno/Administrador)</i>

Reqfun3-Inscripción

Identificador	<i>ReqFun3</i>
Categoría	Funcional
Descripción Corta	<i>Inscripción y Preinscripción del alumno.</i>
Descripción Detallada	<i>El sistema debe brindar la opción al usuario de solicitar una Preinscripción para unirse al Instituto. Dicha preinscripción cuenta de dos partes, la primer parte consiste en completar datos solicitados por la institución e informar al solicitante los distintos requisitos que deberá presentar, de manera presencial en la secretaría, la segunda parte consiste en redireccionar al solicitante al portal ABC para que complete el formulario de preinscripción en dicha página. Una vez hecho todo esto, se generará un Usuario para el Alumno, para su integración con la página.</i>
PreCondiciones	<i>El usuario invitado deberá estar situado en la página de inicio y solicitar Información de la PreInscripción-Inscripción.</i>
Dependencia	<i>Reqfun5--Login (Administrador)</i>
Actor	<i>Usuario (Invitado/Administrador)</i>

Reqf3a--Preinscripción

Identificador	<i>ReqFun3a</i>
Categoría	<i>Funcional</i>
Descripción Corta	<i>Inscripción</i>
Descripción Detallada	<i>Una vez que el solicitante presenta la documentación en secretaría, esta valida la documentación presentada con la información ingresada y valida al solicitante cómo ingresante. (El administrador envía la confirmación)</i>
PreCondiciones	<i>Que se encuentre Pre-inscripto en el sistema y que tenga completado sus datos en ABC.</i>
Dependencia	<i>Reqfun5--Login (Administrador)</i>
Actor	<i>Usuario (Invitado/Administrador)</i>

Reqf3b--Generación de Usuario

Identificador	<i>ReqFun3b</i>
Categoría	<i>Funcional</i>
Descripción Corta	<i>Generacion del usuario (Alumno)</i>
Descripción Detallada	<i>Cuando un administrativo válida a un solicitante, el sistema genera un usuario y una contraseña, únicas para ese usuario, las cuales le servirán para poder acceder a la página y poder realizar dentro de la misma diferentes trámites.</i>
PreCondiciones	<i>Un administrativo válido los datos del usuario</i>
Dependencia	<i>Reqfun5--Login (Administrador)</i>
Actor	<i>Usuario (Invitado/Administrador)</i>

Reqfun4--Noticias

Identificador	<i>ReqFun4</i>
Categoría	Funcional
Descripción Corta	<i>Noticias</i>
Descripción Detallada	<i>El sistema deberá proporcionar al Usuario el apartado de Noticias para su lectura (con sus correspondientes categorías), y la opción de que el mismo usuario pueda compartirla a través de las redes sociales. Mientras que para el Administrador se le debe brindar la opción de que pueda dar de alta, baja y modificar las Noticias.</i>
PreCondiciones	<i>El usuario debe estar parado en la sección de Noticias</i>
Dependencia	Reqfun5--Login (Administrador/Alumno)
Actor	<i>Usuario (Invitado/Alumno/Administrador)</i>

Reqfun5--Login (inicio de sesión)

Identificador	<i>ReqFun5</i>
Categoría	Funcional
Descripción Corta	<i>Login (Inicio de Sesión)</i>
Descripción Detallada	<i>El sistema deberá proporcionar de forma sencilla e intuitiva la capacidad de loguear a un determinado usuario en el el sitio. De manera tal que el mismo tenga acceso a nuevas funcionalidades, las cuales varían, dependiendo del nivel de acceso con el cual cuenta cada usuario.</i>
PreCondiciones	<i>El Usuario debe estar posicionado en la sección de Login para iniciar sesión.</i>
Dependencia	<i>ReqFun3-Inscripción</i>
Actor	<i>Usuario (Alumno/Administrador)</i>

Reqfun5a--Recuperación de contraseña(usuario)

Identificador	<i>ReqFun5a</i>
Categoría	Funcional
Descripción Corta	<i>Recuperación de contraseña (usuario)</i>
Descripción Detallada	<i>El sistema deberá proporcionar un e-mail para validar al usuario que posee una cuenta, y posteriormente se le enviará a su dirección de correo un link para restablecer la contraseña. De esta manera verificamos que el usuario sea el dueño de la cuenta (se desprende del Log-in).</i>
PreCondiciones	<i>El usuario debe estar registrado en el sistema y además posicionado en la sección Log-in y subsección "recuperación de contraseña".</i>
Dependencia	<i>ReqFun3b</i>
Actor	<i>Usuario (Alumno/Administrador)</i>

Reqfun6--Gestión de Trámites

Identificador	<i>ReqFun6</i>
Categoría	Funcional
Descripción Corta	<i>Gestión de Trámites</i>
Descripción Detallada	<i>El sistema deberá tener la capacidad de recibir solicitudes de certificados, historiales de carreras, analíticos, etc. Y devolver un comprobante generado de forma automática, a través de consultas a base de datos.</i>
PreCondiciones	<i>El alumno deberá iniciar una gestión de trámite con el sistema.</i>
Dependencia	<i>Reqfun5--Login (Administrador/Alumno)</i>
Actor	<i>Usuario (Alumno/Administrador)</i>

ReqNOfun7--Diseño Responsive

Identificador	<i>ReqNOfun7</i>
Categoría	<i>No Funcional</i>
Descripción Corta	<i>Diseño Responsive</i>
Descripción Detallada	<i>El sistema deberá proporcionar a todos sus usuarios la opción de ser Responsive (se debe presentar una correcta visualización de la página en todos los dispositivos posibles).</i>
PreCondiciones	<i>El usuario debe estar situado en el sitio</i>
Dependencia	
Actor	<i>Usuario (Invitado/Alumno/Administrador)</i>

Diseño del sistema

Elección de tecnologías:

Al dar comienzo a la planificación del proyecto, surgió como primera incógnita qué lenguajes de programación utilizar y qué recursos destinar al funcionamiento de un servidor web. Finalmente optamos por un stack de desarrollo conformado por Node.js, Express y PostgreSQL.

¿Por qué Node.js?

Node.js es un entorno que nos permite ejecutar código Javascript asíncrono del lado del servidor. Al emplear un modelo de eventos con entrada y salida sin bloqueo, garantiza ser una opción ligera y eficiente, óptima para aplicaciones web en tiempo real. Adicionalmente, nos brinda la posibilidad de emplear el mismo lenguaje de programación (Javascript) tanto para frontend como para backend, simplificando significativamente tareas de planificación, desarrollo e implementación. También, gracias al Node Packet Manager (NPM), podemos obtener miles de módulos actualizados y de código abierto para expandir nuestro desarrollo.

¿Por qué Express?

Express es el framework más popular de Node.js. Suele acompañar casi todos los proyectos hechos en dicho entorno, y su popularidad se debe principalmente a que aporta mecanismos para crear aplicaciones web en menos tiempo y permitir una mayor escalabilidad. Entre sus funcionalidades encontramos enrutamiento con manejadores de peticiones HTTP, gestión de sesiones y cookies, integraciones con motores de renderización de vistas, funciones middleware, entre otras cosas.

¿Por qué Postgresql ?

Una de las principales razones para usar PostgreSQL es que está desarrollado bajo código abierto (Open Source). La capacidad que posee PostgreSQL para correr sobre diferentes Sistemas Operativos nos da la oportunidad de tenerla instalada localmente, realizar una conexión remota a través de su administrador PgAdmin o también desde la aplicación que se esté desarrollando. Desde un punto de vista de acceso a la base de datos el método de *Control de Concurrencias Multiversión* (o por sus siglas en inglés MVCC) ayuda a tener una mejor performance cuando hay muchos movimientos en la base de datos. El principal objetivo de este método es que permite leer y escribir de forma simultánea, es decir, sin que ninguna de las dos operaciones bloquee a la otra. Es muy sencillo de manejar, contamos con herramientas como PgAdmin, esta brinda una interfaz amigable a la hora de interactuar con la base de datos. A su vez podemos encontrar opciones dentro de PgAdmin donde las personas más experimentadas pueden manejar sus códigos específicos abiertamente y sentirse a gusto con ello. Desde un punto de vista de consistencia de los datos cuenta con Hot-Standby, es una de las cualidades más interesantes de Postgres. Además de la facilidad que nos brinda el PgAdmin para hacer mantenimiento de las tablas o respaldos, Hot-Standby permite que los usuarios puedan acceder a las tablas en *modo lectura* mientras que se realizan los procesos de backup o mantenimiento.

¿Por qué Sequelize?

Antes de explicar porque utilizamos sequelize, necesitamos entender qué es un ORM (Object Relational Model). Un *ORM* es un modelo de programación que permite mapear las estructuras de una base de datos relacional (*SQL Server*, *Oracle*, *MySQL*, *Postgres*, etc.), sobre una estructura lógica de entidades con el objeto de simplificar y acelerar el desarrollo de nuestras aplicaciones. Las estructuras de la base de datos relacional quedan vinculadas con las entidades lógicas o *base de datos virtual* definida en el *ORM*, de tal modo que las acciones *CRUD* (*Create*, *Read*, *Update*, *Delete*) a ejecutar sobre la base de datos física se realizan de forma indirecta por medio del *ORM*. La consecuencia más directa es que, además de “mapear”, los *ORMs* tienden a “liberarnos” de la escritura o generación manual de código *SQL* (*Structured Query Language*) necesario para realizar las *queries* o consultas y gestionar la persistencia de datos. Así, los objetos o entidades de la *base de datos virtual* creada en nuestro *ORM* podrán ser manipulados por medio de algún lenguaje de nuestro interés según el tipo de *ORM* utilizado, por ejemplo, *LINQ* sobre *Entity Framework* de *Microsoft*. Con *ORM* nos abstraemos del motor de base de datos que estamos utilizando, es decir, si en algún momento queremos cambiar de base de datos, nos resultará sumamente sencillo. En algunos casos con solo cambiar un par de líneas de código, estaremos cambiando de motor de base de datos.

A continuación explicaremos algunas ventajas y desventajas de la utilizar *ORM*

- Ventajas
 - Facilidad y velocidad de uso.
 - Abstracción de la base de datos usada.
 - Seguridad de la capa de acceso a datos contra ataques.
- Desventajas
 - En entornos con gran carga poner una capa más en el proceso puede mermar el rendimiento.
 - Aprender el nuevo lenguaje del *ORM*.

En este proyecto utilizamos Sequelize, que es un *ORM* para NodeJs, la elección de este *ORM* sobre otros como TypeORM, se redujo a dos factores principales, la primera fue la calidad de la documentación y la segunda a la base de usuarios que cada *ORM* posee. En el caso de ambos *ORM* la documentación presentada en la página oficial era muy detallada y contaba con ejemplos prácticos, con lo cual ambos *ORM* contaban con uno de los factores principales a la hora de utilizar una herramienta, lo cual es contar con una buena documentación. En el segundo factor Sequelize tiene una base de usuarios activos mucho más grande que TypeORM, en consecuencia la cantidad de guías, tutoriales y artículos, relacionados a Sequelize es más extensa lo cual inclinó la balanza hacia Sequelize.

Patrón de diseño MVC

MVC es un **patrón de diseño** que se estructura mediante tres componentes: **modelo, vista y controlador**. Este patrón tiene como principio que cada uno de los componentes esté separado en diferentes objetos, esto significa que los componentes no se pueden combinar dentro de una misma clase. Sirve para clasificar la información, la lógica del sistema y la interfaz que se le presenta al usuario.

- **Modelo:** este componente se encarga de manipular, gestionar y actualizar los datos de una base de datos. No contiene ninguna lógica que describa cómo presentar los datos a un usuario.
- **Vista:** este componente presenta los datos del modelo al usuario. La vista sabe cómo acceder a los datos del modelo, pero no sabe que significa esta información o que puede hacer el usuario para manipularla.
- **Controlador:** este componente se encarga de gestionar las instrucciones que se reciben, atenderlas y procesarlas (la lógica). Por medio del controlador se comunican el modelo y la vista: solicitando los datos necesarios, manipularlos para obtener los resultados y entregarlos a la vista para que pueda mostrarlos.

Los tres componentes de **MVC** están interconectados. La vista muestra el modelo para el usuario, después el controlador acepta la entrada del usuario y actualiza el modelo, y debido a esta acción la vista vuelve a tener un cambio con los datos actualizados.

GIT

Git es un sistema de control de versiones. Un sistema de control de versiones nos va a servir para trabajar en equipo de una manera mucho más simple y óptima cuando estamos desarrollando software. Con Git vamos a poder controlar todos los cambios que se hacen en nuestra aplicación y en nuestro código, y vamos a tener el control absoluto de todo lo que pasa en el código, pudiendo volver atrás en el tiempo, pudiendo abrir diferentes ramas de desarrollo, etc.

Git es un de DVCS (sistema de control de versiones distribuido, por sus siglas en inglés). En lugar de tener un único espacio para todo el historial de versiones del software, como sucede de manera habitual en los sistemas de control de versiones antaño populares, como CVS o Subversion (también conocido como SVN), en Git, la copia de trabajo del código de cada desarrollador es también un repositorio que puede albergar el historial completo de todos los cambios.

Además de contar con una arquitectura distribuida, Git se ha diseñado teniendo en cuenta el rendimiento, la seguridad y la flexibilidad.

Configuraciones iniciales

Antes de empezar el desarrollo de cada uno de los sprint, será necesario establecer una configuración inicial o punto de partida.

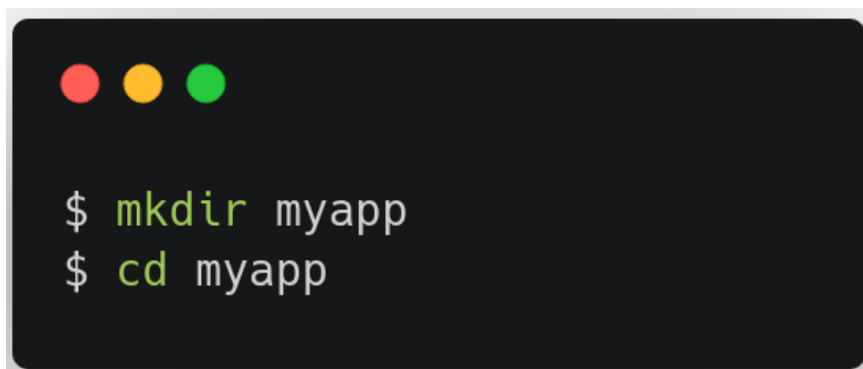
Cómo mencionamos anteriormente utilizamos NodeJS como entorno en tiempo de ejecución multiplataforma, para la capa del servidor, lo que nos permite ejecutar código JavaScript fuera del navegador. Node brinda de manera nativa varias funciones para poder manejar peticiones. Estas funcionalidades que nos brinda Node son muy elementales, por tal motivo, utilizamos un framework conocido cómo Express.

Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web. Cuenta con miles de métodos de programa de utilidad HTTP y middleware. Express además proporciona una delgada capa de características de aplicación web básicas, que no ocultan las características de NodeJs, por lo tanto funciona cómo una interfaz entre NodeJs y el programador.

Partiremos asumiendo la instalación de NodeJs.

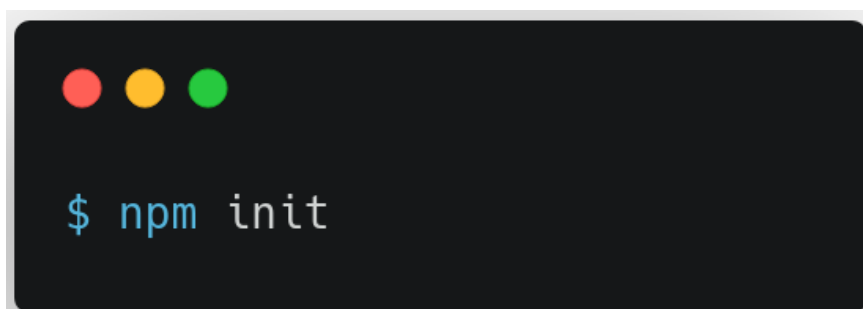
Instalación y configuración de Express

Suponiendo que ya ha instalado [Node.js](#), cree un directorio para que contenga la aplicación y conviértalo en el directorio de trabajo.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The terminal shows two commands being executed: '\$ mkdir myapp' and '\$ cd myapp', both with green text for the command names.

```
$ mkdir myapp
$ cd myapp
```

Utilice el mandato npm init para crear un archivo package.json para la aplicación.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The terminal shows the command '\$ npm init' being executed, with 'npm' in blue and 'init' in white text.

```
$ npm init
```

Este mandato solicita varios elementos como, por ejemplo, el nombre y la versión de la aplicación. Por ahora, sólo tiene que pulsar INTRO para aceptar los valores predeterminados para la mayoría de ellos, con la siguiente excepción:



```
Press ^C at any time to quit.  
package name: (myapp)  
version: (1.0.0)  
description:  
entry point: (index.js)
```

Especifique app.js o el nombre que desee para el archivo principal. Si desea que sea index.js, pulse ENTER para aceptar el nombre de archivo predeterminado recomendado.

A continuación, instale Express en el directorio myapp y guárdelo en la lista de dependencias. Por ejemplo:



```
$ npm install express --save
```

Para instalar Express temporalmente y no añadirlo a la lista de dependencias, omita la opción --save:



```
$ npm install express
```

Configuración de PostgreSQL

A la hora de interactuar con la base de datos, es necesario la existencia de un usuario. En un principio resulta sencillo y cómodo hacer uso del usuario por defecto proporcionado por Postgres. Sin embargo, en nuestro análisis del sistema identificamos una variedad de usuarios con diferentes privilegios ya sea de lectura, escritura, creación o destrucción de la información que es almacenada en la base de datos. Por este motivo, decidimos atacar esta problemática a través de los GRUPOS,

Técnicamente podríamos crear usuarios y asignar ROLES a cada usuario, pero debemos considerar que el sistema, a nivel usuarios, no es constante. Esto quiere decir que los usuarios van *cambiando* a lo largo del tiempo. Desde un punto de vista administrativo resulta mucho más eficiente crear GRUPOS con privilegios, en vez de asignarle a cada usuario privilegios. Al utilizar la filosofía de crear grupos con privilegios, se obtienen las siguientes ventajas:

- Permite generar una jerarquía entre los diferentes tipos de usuarios con lo que se cuenta.
- Es mucho más sencillo eliminar un usuario del sistema, ya que este no tiene privilegios.
- Es más eficiente a agregar/remover usuarios de un grupo, que otorgar o revocar privilegios cada vez que se agrega un usuario o cambian los privilegios del mismo.

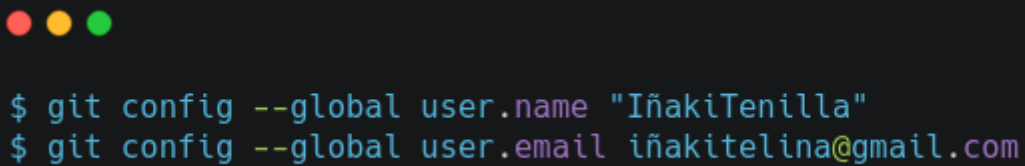
La desventaja principal de esta solución es que eleva la complejidad de la base de datos, es decir ahora no solamente debemos garantizar la consistencia de los datos almacenados, sino que también tendremos que asegurar que los usuarios solamente puedan acceder a los datos que tengan permitidos.

Configuración de Git

En el siguiente apartado detallaremos la configuración inicial de git así como algunos de los comandos básicos utilizados a lo largo del proyecto.

Configuraciones iniciales

En primera instancia deberemos configurar tanto nuestro usuario como un email, el cual se utilizara para identificar al usuario. Estos datos se utilizarán para identificar al usuario dentro del repositorio. Es importante notar que estos datos no se utilizan como credenciales para los repositorios remotos.



```
$ git config --global user.name "IñakiTenilla"  
$ git config --global user.email iñakitelina@gmail.com
```

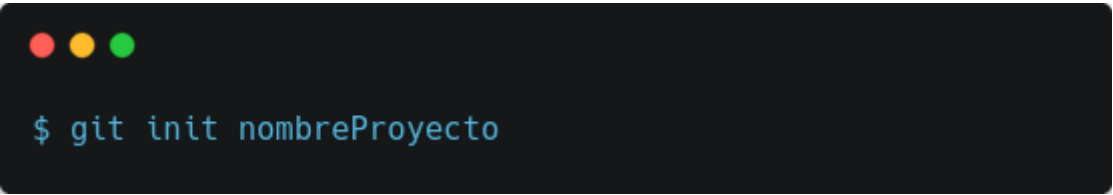
Inicialización de un nuevo Repositorio Local

Inicializa un repositorio dentro de la carpeta :



```
$ git init
```

También contamos con la opción de crear una carpeta e inicializar un repositorio local mediante:



```
$ git init nombreProyecto
```

Comandos básicos:

Status:


Informa es estado del repositorio actual, si se realizó alguna modificación o si hay alguna modificación respecto de algún cambio que se encuentra en seguimiento:



```
$ git status
```

Add:

Genera un punto de salvado potencial con este comando, se pone en seguimiento el archivo "nombreArchivo", con el comando git status se puede ver que el cambio dejó de estar en rojo y pasó a estar en verde.



```
$ git add nombreArchivo
```

Este comando tiene una segunda forma de ser usado y es en vez de pasarle un nombre pasarle un ".", de esta forma se agrega al seguimiento todos los archivos que fueron modificados (estado en rojo) .



```
$ git add .
```

Commit:


El commit es el guardado definitivo permite empaquetar un conjunto de cambios y guardarlos en el repositorio. SOLAMENTE va a guardar los cambios que están en seguimiento (los que al utilizar git status se vean en verde).



```
$ git commit -m "Mensaje corto" -m "Mensaje detallado"
```

Checkout

Es un comando que nos permite realizar varias acciones. La primera función y la más usada es como CTRL Z nos va permitir deshacer un cambio. ¿Con qué criterio? En primera instancia va a utilizar los archivos de seguimiento, que son los que agregamos con el git add como punto de salvado potencial, en caso de que no tengamos un punto de salvado agregado con el git add tomará como referencia el último commit realizado. Lo utilizamos de la siguiente manera:



```
$ git checkout -- nombreArchivo
```

Log:

Para ver todos los commits que realizamos podemos usar el comando:



```
$ git log
```


El comando log admite varios parámetros, uno de los más usado sería oneline , este nos va a mostrar el hash del commit y el mensaje corto del mismo.



```
$ git log --oneline
```

Commit --amend

Puede suceder que realizamos un commit y nos olvidamos de agregar archivos o el nombre que le asignamos no era el correcto. En este caso, para poder modificar el último commit podemos utilizar el comando:

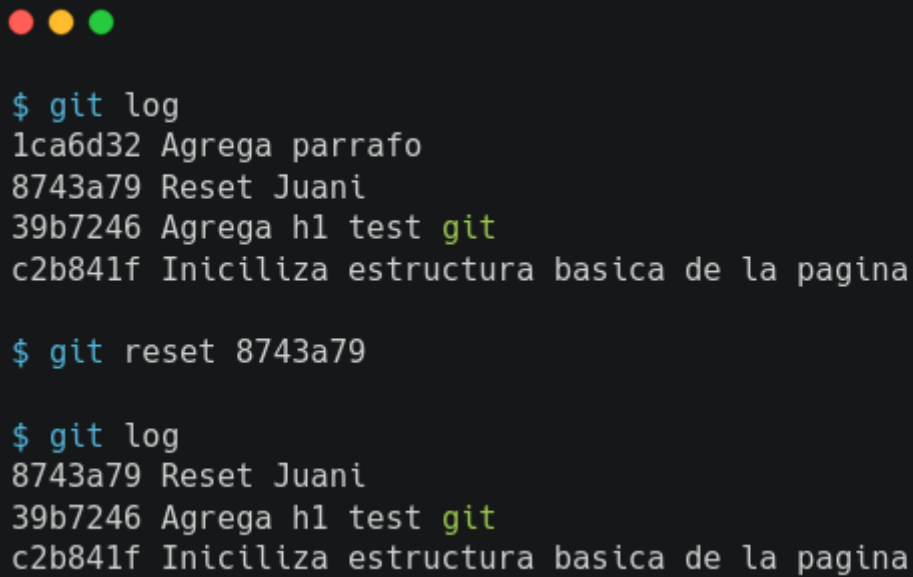


```
$ git commit --amend
```

Cuando ejecutemos el comando todo lo que se encuentre en seguimiento, es decir archivos que agregamos con el git add Archivo, pasarán a formar parte del commit. Y nos permitirá también modificar el nombre del commit.

Reset

Este comando va a destruir los commits, hasta el Nrocommit, que le pasemos al comando:



```
$ git log
1ca6d32 Agrega parrafo
8743a79 Reset Juani
39b7246 Agrega h1 test git
c2b841f Iniciliza estructura basica de la pagina

$ git reset 8743a79

$ git log
8743a79 Reset Juani
39b7246 Agrega h1 test git
c2b841f Iniciliza estructura basica de la pagina
```

Por defecto va a dejar en el área de trabajo los cambios, es decir si hacemos un **git status** vamos a ver los cambios del **commit Agrega párrafo** en rojo.

Es posible destruir más de un commit, pero no es recomendable hacerlo ya que puede llegar a generar conflictos a nivel local y peor aún al momento de sincronizar con un repositorio remoto.

Git reset tiene dos parámetros:

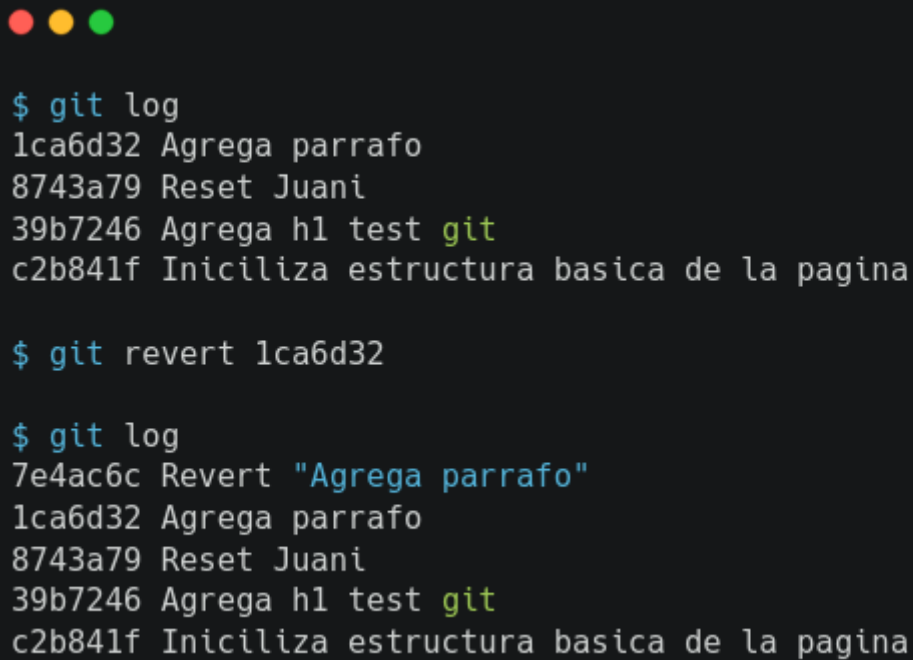
- *soft* en caso de usar este parámetro, destruirá el commit y dejará en seguimiento los cambios de ese commit (es decir si hacemos git status los cambios realizados en el commit estarán en verde), aunque destruyendo el commit en el proceso.
- *hard* al utilizar este parámetro elimina el commit y descarta todos los cambios realizados, dejando el directorio de trabajo en el mismo estado que el commit que se hizo referencia.



```
$ git reset --soft NroCommit
$ git reset --hard NroCommit
```

Revert

El comando revert realiza la operación opuesta al commit que se le envía.



```
$ git log
1ca6d32 Agrega parrafo
8743a79 Reset Juani
39b7246 Agrega h1 test git
c2b841f Iniciliza estructura basica de la pagina

$ git revert 1ca6d32

$ git log
7e4ac6c Revert "Agrega parrafo"
1ca6d32 Agrega parrafo
8743a79 Reset Juani
39b7246 Agrega h1 test git
c2b841f Iniciliza estructura basica de la pagina
```

En el ejemplo anterior tomo el commit que agregaba un párrafo, y genera un nuevo commit realizando la operación inversa, es decir, si en el commit Agregar párrafo se agregan líneas, en el commit Revert "Agregar párrafo" se quitan las líneas.

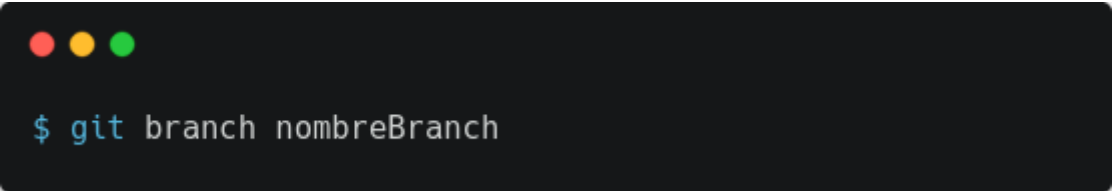
Branch:

Para ver las ramas con las cuentas de nuestro proyecto ingresamos el siguiente comando:



```
$ git branch
```

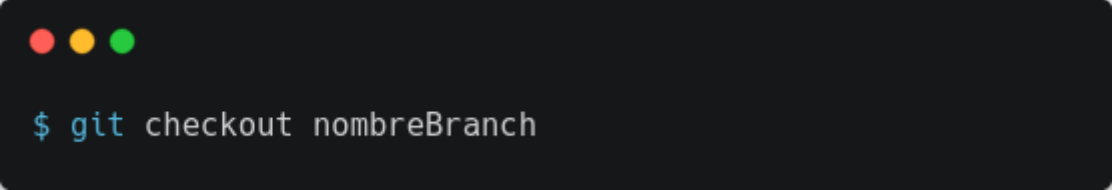
Al usar git branch nombreRama estamos generando una nueva "versión" o "rama" de nuestro proyecto:



```
$ git branch nombreBranch
```


Checkout

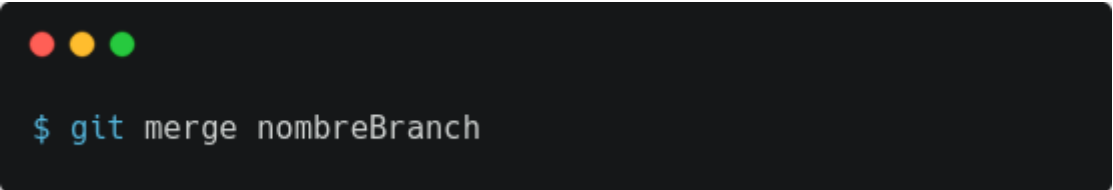
Para moverse entre ramas podemos usar el comando checkout:



```
$ git checkout nombreBranch
```

Merge:

Para unir ramas distintas se usa el comando git merge seguido del nombre de la rama que quieres unir, si no ocurre ningún conflicto el merge genera un commit con la fusión de las dos ramas.

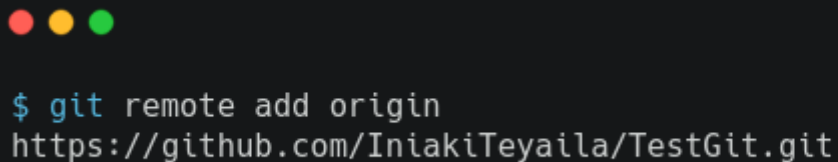


```
$ git merge nombreBranch
```

Remote:

Con el comando push podemos "empujar" nuestro repositorio a un repositorio remoto. En este caso utilizamos GitHub para pushear nuestro repositorio. Pero primero tendremos que configurar los remotos de manera local.

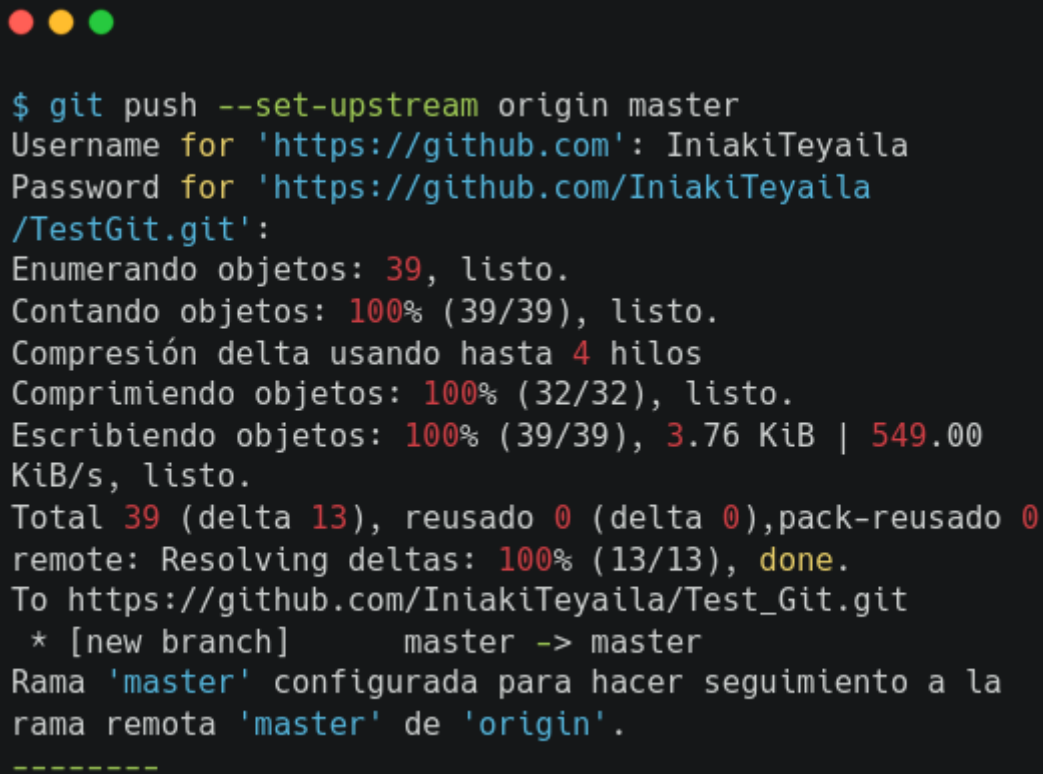
Para esto necesitamos crear un nuevo repositorio dentro de GitHub , una vez que lo tenemos creado, necesitamos agregar en nuestro repositorio local el remoto al cual queremos vincular con nuestro repo local con el siguiente comando:



```
$ git remote add origin  
https://github.com/IniakiTeyaila/TestGit.git
```

Push

Luego tendremos que setear un upStream hacia el repo remoto de la siguiente manera :



```
$ git push --set-upstream origin master
Username for 'https://github.com': IniakiTeyaila
Password for 'https://github.com/IniakiTeyaila/TestGit.git':
Enumerando objetos: 39, listo.
Contando objetos: 100% (39/39), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (32/32), listo.
Escribiendo objetos: 100% (39/39), 3.76 KiB | 549.00 KiB/s, listo.
Total 39 (delta 13), reusado 0 (delta 0), pack-reusado 0
remote: Resolving deltas: 100% (13/13), done.
To https://github.com/IniakiTeyaila/Test_Git.git
 * [new branch]      master -> master
Rama 'master' configurada para hacer seguimiento a la
rama remota 'master' de 'origin'.
-----
```

Al final podremos empujar nuestro repositorio hacia GitHub, cada vez que sufra modificaciones a nivel local.

Pull:

Para poder sincronizar los cambios que fueron subidos a GitHub utilizamos el comando pull, que básicamente trae los cambios desde el repositorio remoto.

```
$ git pull
```

```
warning: Hacer un pull sin especificar cómo reconciliar  
las ramas es poco  
recomendable. Puedes eliminar este mensaje usando uno  
de los  
siguientes comandos antes de tu siguiente pull:
```

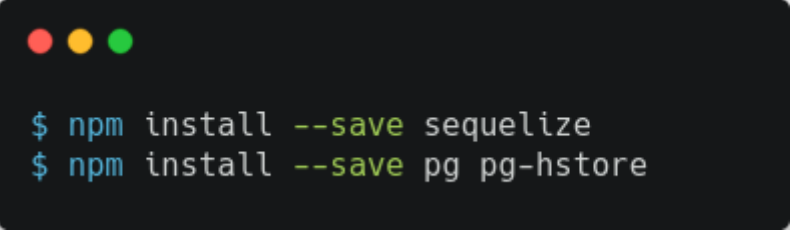
```
git config pull.rebase false # hacer merge  
(estrategia por defecto)  
git config pull.rebase true  # aplicar rebase  
git config pull.ff only      # aplicar solo fast-  
forward
```

Puedes reemplazar "git config" con "git config --global" para aplicar la preferencia en todos los repositorios. Puedes también pasar --rebase, --no-rebase, o --ff-only en el comando para sobrescribir la configuración por defecto en cada invocación.

Ya está actualizado.


Instalación y configuración de sequelize

Para instalar sequelize en nuestro proyecto de node solo tenemos que escribir las siguientes líneas en la consola:



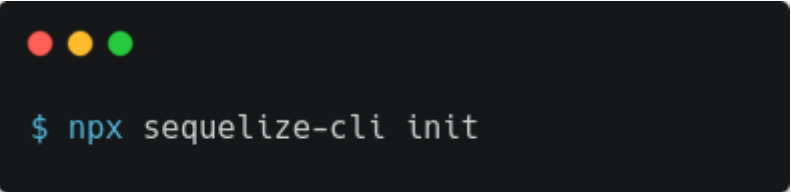
```
$ npm install --save sequelize  
$ npm install --save pg pg-hstore
```

Adicionalmente instalaremos el CLI de sequelize.



```
$ npm install --save-dev sequelize-cli
```

Sequelize-cli es una interfaz de línea de comando la cual nos permite crear base de datos, generar modelos, migraciones y sembrado de datos usando la terminal. Para inicializar el cli utilizamos el siguiente comando.



```
$ npx sequelize-cli init
```

Este comando generará las carpetas dentro de nuestro directorio de trabajo, dentro de las cuales se almacenarán los modelos, semillas y archivos de configuración relacionados a la base de datos y al funcionamiento específico del CLI.

Una vez instalado sequelize-cli podemos utilizar el siguiente comando para obtener una lista de los comandos disponibles.

```
$ npx sequelize-cli --help
npx: instaló 80 en 45.029s

Sequelize CLI [Node: 15.2.1, CLI: 6.2.0, ORM: 6.3.5]

sequelize <command>

Comandos:
  sequelize db:migrate           Run pending migrations
  sequelize db:migrate:schema:timestamps:add  Update migration table to have timestamps
  sequelize db:migrate:status    List the status of all migrations
  sequelize db:migrate:undo      Reverts a migration
  sequelize db:migrate:undo:all  Revert all migrations ran
  sequelize db:seed              Run specified seeder
  sequelize db:seed:undo         Deletes data from the database
  sequelize db:seed:all          Run every seeder
  sequelize db:seed:undo:all     Deletes data from the database
  sequelize db:create            Create database specified by configuration
  sequelize db:drop              Drop database specified by configuration
  sequelize init                 Initializes project
  sequelize init:config          Initializes configuration
  sequelize init:migrations      Initializes migrations
  sequelize init:models          Initializes models
  sequelize init:seeders         Initializes seeders
  sequelize migration:generate   Generates a new migration file
                                  [alias: migration:create]
  sequelize model:generate       Generates a model and its migration
                                  [alias: model:create]
  sequelize seed:generate        Generates a new seed file
                                  [alias: seed:create]

Opciones:
  --version  Muestra número de versión  [booleano]
  --help     Muestra ayuda                [booleano]
-----
```

Generación de modelos

Para generar los modelos por medio del CLI utilizamos:

```
npx sequelize-cli model:generate --name User
--attributes firstName:string,lastName:string,email:string,password:string
```

En este ejemplo estamos generando un modelo User con atributos firstName, lastName, email, password de tipo String. Lo cual generará dentro de la carpeta src en la subcarpeta models un nuevo modelo con la siguiente estructura:

```
'use strict';
module.exports = (sequelize, DataTypes) => {
  const User = sequelize.define('User', {
    firstName: DataTypes.STRING,
    lastName: DataTypes.STRING,
    email: DataTypes.STRING,
    password: DataTypes.STRING,
  }, {});
  User.associate = function(models) {
    // associations can be defined here
  };
  return User;
};
```

En este modelos tendrán que ir las asociaciones según fueron modeladas en el diagrama relacional. Cada atributo del modelo User tiene un DataType o un JSON, mediante el cual se puede ser más específico a la hora de detallar las características de dicho atributo (si es o no llave primaria, si permite null, restricciones sobre los datos que almacenan, etc.), para más información en el apartado de bibliografía se adicionan links a la documentación y a diferentes artículos, los cuales fueron utilizados para tener una mejor comprensión del funcionamiento y las capacidades de esta características.


Generación de migración

Las migraciones se utilizan para vincular, a nivel estructural, los modelos del lado del backend con las tablas que se almacenan en la base de datos, generando las tablas en caso de que estas no existan.

Esto nos permite reestructurar/migrar la estructura básica de nuestra base en cualquier otra base de datos de otro tipo (soportada por el ORM). Una de las funciones más interesante de las migraciones es que nos permite modificar de manera sencilla la estructura de nuestra base de datos, además nos brinda una forma de control de versiones sobre las modificaciones realizada a las tablas, desde un punto de vista estructural y no de los datos que se tiene almacenados.

Las migraciones se trabajan en forma de pila, esto quiere decir que al momento de deshacer migraciones, estas se irán revirtiendo desde la más nueva hacia la más antigua. Al generar un modelo se crea también de manera automática una migración asociada a dicho modelo, sequelize asume que el modelo que fue generado no se encuentra en la base de datos, aunque no siempre será el caso.

Para generar una migración de manera manual utilizamos el siguiente comando.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The command being entered is in a light blue color.

```
$ npx sequelize-cli migration:generate --name User
```

La estructura básica de una migración asociada al modelo será la siguiente

```
use strict';
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('Users', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      firstName: {
        type: Sequelize.STRING
      },
      lastName: {
        type: Sequelize.STRING
      },
      email: {
        type: Sequelize.STRING
      },
      password: {
        type: Sequelize.STRING
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: (queryInterface, Sequelize) => {
    return queryInterface.dropTable('Users');
  }
}
```


Sembrado de modelos

Sequelize permite la carga de datos mediante semillas, la finalidad de esta funcionalidad es la carga de grandes volúmenes de datos.

Mediante el siguiente comando se puede crear una estructura básica de una semilla:

```
$ npx sequelize-cli seed:generate --name Rol
```

El comando anterior nos genera una estructura muy similar a la detallada a continuación.

```
'use strict';

module.exports = {
  up: async (queryInterface, Sequelize) => {
    var rol=[{
      "nombreRol": "Directivo"
    }, {
      "nombreRol": "Administrativo"
    }, {
      "nombreRol": "UsuarioRegistrado"
    }, {
      "nombreRol": "AdministradorPagina"
    }]
    await queryInterface.bulkInsert('Rol',rol
    , {});
  },

  down: async (queryInterface, Sequelize) => {
    await queryInterface.bulkDelete('Rol', null, {});
  }
};
```

Mackaroo

Para la generación de datos utilizamos la página **mackaroo**, la cual nos permite generar esquemas de tablas y dataset (rangos de datos o enumerados) y así generar datos de manera mucho más dinámica.

Esquemas

My Schemas

Create a SchemaImport from File...Delete Selected

Schema	Last Modified		
AlumnoPendiente fotoCarnet certificadoSalud fotocopiaDni nombre apellido fechaNac dni email direccion numTelefono abcFormulario	about 1 month ago		<input type="checkbox"/>
Carrera nombre nombreAbreviado titulo resolucion duracion horas tipo modalidad valida	about 1 month ago		<input type="checkbox"/>
ConsultaFormulario email nombre consulta	about 1 month ago		<input type="checkbox"/>
Matrías nombreProfesor cargaHoraria contenido dias	about 1 month ago		<input type="checkbox"/>
Noticia tituloNoticia cuerpoNoticia visibilidad	about 1 month ago		<input type="checkbox"/>
Rol nombreRol	about 1 month ago		<input type="checkbox"/>
Tramites tipoTramite fechaInicio estado fechaFinalizacion token	about 1 month ago		<input type="checkbox"/>
Usuarios id first_name last_name email gender ip_address	5 months ago		<input type="checkbox"/>

DataSet

My Datasets

Have existing data you want reference in your schemas?
Upload your CSV datasets here, then import columns into your schemas using the Dataset Column type.

Upload a New DatasetDelete Selected

Name	File	Last Modified		
cargaHoraria	cargaHoraria.csv	about 1 month ago		<input type="checkbox"/>
Carreras	carreras.csv	about 1 month ago		<input type="checkbox"/>
categoria	categoria.csv	about 1 month ago		<input type="checkbox"/>
DiasLaborables	dias.csv	about 1 month ago		<input type="checkbox"/>
nombreRol	nombreRol.csv	about 1 month ago		<input type="checkbox"/>
tipoTramite	tipoTramite.csv	about 1 month ago		<input type="checkbox"/>

Instalación y configuración de Passport

Adicionalmente, instalaremos **Passport.js**, el cual se describe a sí mismo como un middleware de autenticación *<simple y discreto>* para Node.js. Haremos uso de sus funcionalidades para administrar las sesiones dentro del proyecto.

```
$ npm install passport express-session cookie-parser --save
```

Con el mismo comando, agregaremos las dependencias **cookie-parser** y **express-session**. Cookie-parser es otro middleware cuya función principal está referida al manejo, escritura y lectura de cookies. Mientras que express-session almacena los datos de sesión en el servidor, guardando solo el ID de sesión de la propia cookie. Ambos son considerados por los desarrolladores de express como buenas prácticas de seguridad.

```
const passport = require('passport');
const cookieParser = require('cookie-parser');
const session = require('express-session');
```

Las tres dependencias deben ser llamadas desde el archivo de configuración de node.js, siendo también requeridas en archivos que necesiten sus funcionalidades.

```
app.use(cookieParser(process.env.COOKIE_SECRET));

app.use(session({
  secret: process.env.COOKIE_SECRET,
  resave: true,
  saveUninitialized: true
}));
```

Una vez realizadas las llamadas a las dependencias en cuestión, inicializamos cookieParser estableciendo el secreto. Su valor estará referenciado por una variable de entorno llamada COOKIE_SECRET.

```
app.use(passport.initialize());
app.use(passport.session());
```

La función `initialize()` activa `passport`, mientras que `session()` es un middleware que mantiene el valor “usuario” referenciado al ID actual de sesión.

Passport utiliza lo que denomina “estrategias” para establecer el tipo de autenticación. Existen estrategias para autenticarse con cuentas de google, facebook, twitter, github, etcétera. En este caso, emplearemos la autenticación local. Para ello, debemos añadir el siguiente módulo:

```
$ npm install passport-local --save
```

Una vez instalado, lo importamos:

```
const PassportLocal = require('passport-local').Strategy;
```

La autenticación se lleva a cabo invocando la estrategia y declarando una función con tres argumentos: `username`, `password` y `done`. La función buscará en la base de datos el usuario correspondiente. En caso de que no exista o la contraseña sea incorrecta, el callback “done” será devuelto como falso. En caso de que las credenciales sean correctas, se devolverá el usuario.

```
passport.use(new PassportLocal(
  function(username, password, done) {
    User.findOne({ username: username }, function (err, user) {
      if (err) { return done(err); }
      if (!user) { return done(null, false); }
      if (!user.verifyPassword(password)) { return done(null,
false); }
      return done(null, user);
    });
  }
));
```

```
app.post('/login',  
  passport.authenticate('local', { failureRedirect: '/login' } ),  
  function(req, res) {  
    res.redirect('/');  
  });
```

Entre los enrutamientos, podemos definir la autenticación cuando se active un método HTTP “Post” en la ruta /login. Esto activará la función authenticate, que entre sus argumentos establece la estrategia (en este caso, local), la dirección de redirección en caso de que las credenciales (o lo que fuera) falle, y por último una función donde se establece la URL de redirección en caso de que el login sea exitoso.

```
passport.serializeUser(function(user, done) {  
  done(null, user.id);  
});  
  
passport.deserializeUser(function(id, done) {  
  User.findById(id, function(err, user) {  
    done(err, user);  
  });  
});
```

Implementación del proyecto

VISTA PRINCIPAL



Esta es la primera parte de la vista principal.

En la cabecera (barra azul/violeta) tenemos los datos de contacto (teléfono y mail) bien a la vista porque son de las cosas que más "a mano" tiene que tener el visitante.

Abajo a la izquierda está el logo del ISFT, usa formato PNG para tener fondo transparente.

A la derecha están los botones que tienen color blanco (en el resto de secciones es negro, se optó por blanco para quedar bien con la mayoría de imágenes).

El carrusel está hecho con una librería llamada *revolution slider* (documentación en la página oficial de la librería), se tomó la configuración por defecto. Esta librería nos permite agregarle "capas" (layers) al carrusel, donde cada capa es un elemento como texto (por ej: periodo de inscripción) o un botón (o conjunto de botones). A cada capa se le puede asignar color, tamaño, coordenada X e Y dentro del div del carrusel, se le puede agregar animaciones y tiempos de las animaciones, también se puede agregar texto, botones, etc. sobre la imagen en cuestión.

La idea si fuera una implementación a futuro, es que las imágenes y textos del carrusel sean actualizables por el administrativo desde su panel de control, para que siempre esté en primera plana la información más nueva y relevante.

Las imágenes que usa el carrusel son 1920 x 800, sin embargo, el carrusel tiene la capacidad de redimensionar las fotos perdiendo muy poca calidad.

El carrusel no está conectado en el Back-End, está todo hardcodeado en el código, más adelante en una futura implementación se puede conectar al Back.

SECCIÓN CARRERAS



Más abajo se encuentran las carreras, se conectan directamente a la tabla "carreras" de la base de datos y extrae la información de ahí.

El título que se muestra es el NOMBRE ABREVIADO de la carrera. Lo que quiere decir, es que en el formulario de creación de carrera, el administrativo agrega el nombre completo ("Técnico Superior en Análisis de Sistemas"), el nombre abreviado ("Análisis de Sistemas") y el título ("Analista de Sistemas"), entre esas 3 opciones, nuestro sistema selecciona el nombre abreviado para mostrarlo como nombre de carrera, ya que es más práctico/corto.

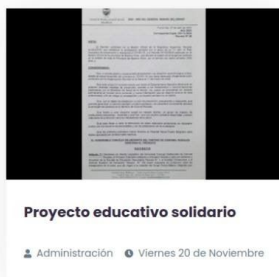
La imagen es subida por el administrativo desde un formulario, y no importa si es PNG y el tamaño es 2000x2000 o cualquiera, SIEMPRE se va a redimensionar a JPG y un tamaño fijo de 600x300 (esto a través de la librería "SHARP" que funciona en el backend).

Tratamiento de imágenes, consideraciones:

- Se opta por JPG porque es el formato de imagen comprimido (más liviano), PNG está sin comprimir y pesa más.
- Se prioriza por usar el mismo tamaño para que todo esté *simétrico* y la imagen se redimensiona de modo **proporcional** (se busca el mejor ratio ancho/largo) para que no pierda calidad la imagen.
- El resto (título de la carrera, resolución, duración, carrera vigente o no) está completamente funcional y se extrae desde el backend.

SECCIÓN NOTICIAS

Noticias



El tratamiento de imágenes para las noticias se gestiona de la misma manera que las imágenes de la sección carreras, la principal diferencia es que el JPG, se redimensiona a 370 x 230.

Cuando la imagen no se puede redimensionar de manera que ocupe todo el ancho deseado del div (o otro elemento html), se le agrega el fondo negro que se ve en la imagen superior. La calidad no se pierde.

Cuando se pasa el mouse por la noticia aparece parte del cuerpo de la noticia con el efecto hover, todo funcional y conectado al backend.

INGRESANDO A LA NOTICIA

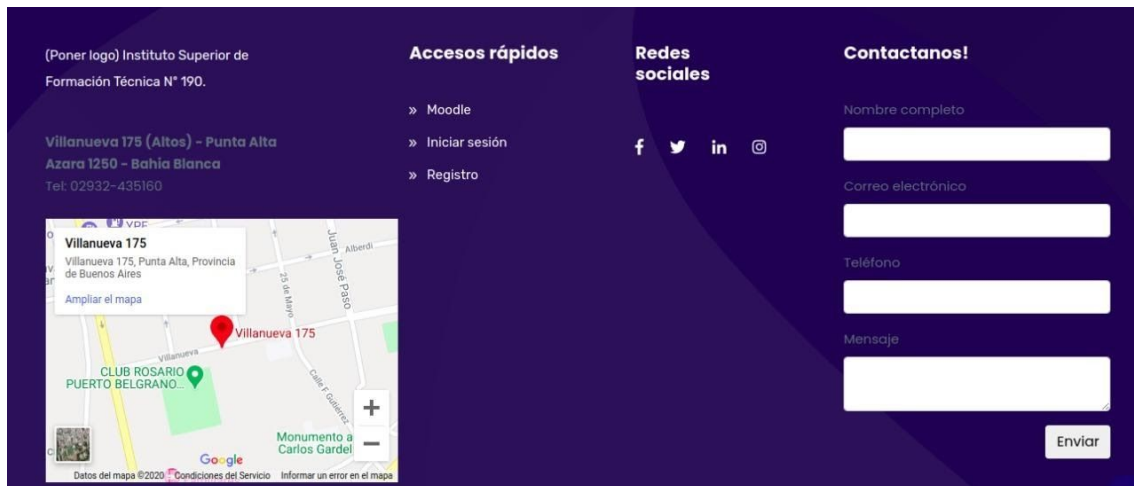


Al clickear una noticia aparecería la vista de la misma, donde figura el título, quién lo publicó, la fecha y la categoría. A la derecha aparece un newsletter (no funcional, se podría implementar a futuro) para que las noticias lleguen automáticamente al correo del inscripto.

A la izquierda aparecen botones para compartir en las redes sociales más importantes (Facebook, Twitter, LinkedIn y Instagram).

La primera foto que aparece es la que el administrativo subió primero, luego figura el texto, y por último las otras fotos que subió el administrativo.

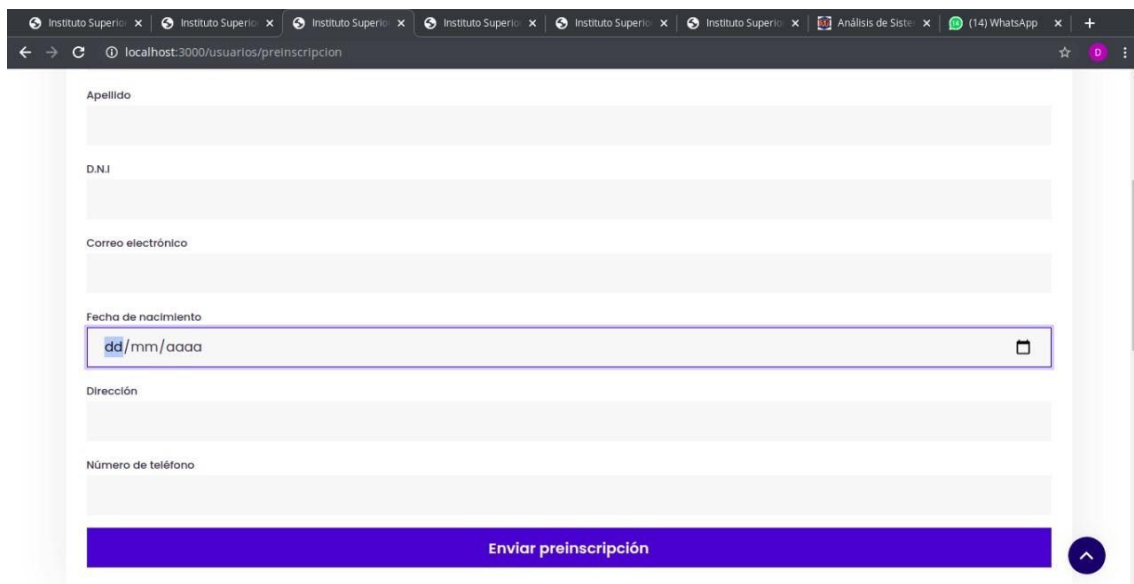
SECCIÓN FOOTER



En el footer abajo a la izquierda se usa un IFRAME (Un **Iframe** es un documento de una web que se inserta en otra página web) de google maps, con la dirección actual del instituto y funcional.

Después son todos botones de acceso rápido y a redes sociales, y un formulario de contacto que se conectaría con el correo electrónico del Instituto (en un futuro).

FORMULARIO DE PREINSCRIPCIÓN



El formulario de pre-inscripción es igualito a los demás, es un formulario que se comunica con la tabla "usuarios pendientes" de la base de datos, se carga con datos personales.


Lo único diferente acá es que se usa un campo tipo DATE (fecha), este campo es una interfaz gráfica para elegir tu fecha de nacimiento con el mouse, también es posible ingresarla manualmente.

Empleamos como framework principal Bootstrap.

Una vez que se envía la preinscripción, el administrativo va a poder ver desde su panel de control aquellos usuarios pendientes y a su vez tendrá la opción de confirmarlos, editarlos y borrarlos.

SECCIÓN LOGIN

(02932) 435-160 isft190estudiantes@gmail.com



INICIO CONOCENOS NOTICIAS Y EVENTOS MOODLE CONTACTO **INGRESAR**

Ingresar

Acceder

☐ No cerrar sesión.

[¿Olvidaste tu contraseña?](#)

[Inscripción](#)


La

Vista de login, permite a un usuario, alumno o administrativo loguear en el sistema.

MENU USUARIO

(02932) 435-160 isft190estudiantes@gmail.com

Bienvenido, **Chrotoem Teager** **Panel de control**



INICIO CONOCENOS NOTICIAS Y EVENTOS MOODLE CONTACTO DESCONECTARME

Panel de control administrativo/estudiantil

Datos de usuario

Nombre de usuario:

Nombre completo:

Rol:

cteager1

Chrotoem Teager

Estudiante

Menú de acciones estudiantiles

Solicitar constancia de alumno

Consultar inscripciones a mesas de examen

Análisis en línea

Contactar

Chrotoem Teager

Cuando ingresamos con un usuario (en este caso un estudiante, pero es casi lo mismo para todos) en la cabecera arriba a la derecha nos va a "saludar" con nuestro nombre, apellido y un botón que nos redirecciona a nuestro perfil/panel de control donde aparecen los datos básicos. Sobre la derecha se puede visualizar un formulario de contacto directo con administración.

Dependiendo del rol se despliega un menú de acciones. en el caso del alumno se puede generar una constancia, solicitar notas, inscribirse a finales, etc.

GENERAR CONSTANCIA DE ALUMNO REGULAR

The screenshot shows a web application interface. At the top left is a logo with the letters 'ISFA'. A modal window titled 'Generar constancia de alumno' is open in the center. It contains a dropdown menu labeled 'A ser presentado a:' with 'Autoridades IOSFA' selected. Below the dropdown are two buttons: 'Cerrar' (Close) and 'Generar' (Generate). The background shows a user profile section with the name 'Chrotoem Teager' and a 'Contactar' section with an email input field and a message input field.

Al generar una constancia, se abre una ventana (denominada MODAL en lo que es Bootstrap y el diseño web en general), donde se aclara el destino al cual será presentada la constancia.

CONSTANCIA

The image shows a generated certificate titled 'CONSTANCIA DE ALUMNO REGULAR'. It is issued by the 'DIRECCIÓN GENERAL DE CULTURA Y EDUCACIÓN' of the 'GOBIERNO DE LA PROVINCIA DE BUENOS AIRES'. The certificate states that 'Chrotoem Teager, DNI 25.523.127' is a student of the 'Instituto Superior de Formación Técnica Nro. 190 de Coronel Rosales' in the 'Técnico Superior en Análisis de Sistemas' career. It references 'Res. 5817/03' and states that the certificate is valid for the 'ciclo lectivo 2020'. The certificate includes a QR code on the bottom left, a circular official seal on the bottom right, and a digital signature.

Al apretar generar constancia, genera un certificado hecho en HTML + CSS, que fácilmente se puede convertir a PDF u otro formato usando una librería. Se agregan imágenes del ministerio y abajo a la derecha la firma digitalizada.


Hay un código QR que serviría para validar si la constancia.

Los datos que se muestran en este momento son estáticos, no existe una vinculación con la base de datos del instituto.

MENU ADMINISTRATIVO

(02932) 435-160
istft190estudiantes@gmail.com

Bienvenido, **Nappie Dimmne**
Panel de control



INICIO
CONOCENOS
NOTICIAS Y EVENTOS
MOODLE
CONTACTO
DESCONECTARME

Panel de control administrativo/estudiantil

Datos de usuario

Nombre de usuario: ndimmne0
Nombre completo: Nappie Dimmne
Rol: Administrador

Menú de acciones administrativas

Consultar listado completo de alumnos
Consultar preinscripciones pendientes

Contactar

Nappie Dimmne

Email*


Mensaje

Esta vista corresponde al perfil del administrador, es similar a la del alumno pero esta contiene otras 2 acciones distintas. Una es consultar usuarios ya existentes, otra es consultar preinscritos.

LISTADO DE ALUMNOS PREINSCRITOS

(02932) 435-160
istft190estudiantes@gmail.com

Bienvenido, **Nappie Dimmne**
Panel de control



INICIO
CONOCENOS
NOTICIAS Y EVENTOS
MOODLE
CONTACTO
DESCONECTARME

Listado de alumnos en proceso de inscripción.

Buscar...

N° Preinscripción	Nombre	DNI	Fecha de nacimiento	Correo electrónico	Domicilio	Número de teléfono	Foto carnet	Certificado de salud	Fotocopia DNI	Acciones
1	Gino Rossi	42133081	Wed Sep 15 1999 21:00:00 GMT-0300 (hora estándar de Argentina)	ginorossiar@gmail.com	8° Centenario Casa N° 209	No	No	No	No	Confirmar Editar Borrar
2	Agustin Rodriguez	43584203	Tue Jun 25 2019 21:00:00 GMT-0300 (hora estándar de Argentina)	biablabl@outlook.com.ar.es	cualquiera	No	No	No	No	Confirmar Editar Borrar

El administrativo puede ver desde la vista los alumnos preinscritos (esto está directamente conectado a la tabla "alumnosPendientes"), puede ver su número de preinscripción, nombre, dni, fecha nacimiento, correo, dirección, etc.

Puede filtrar todas las columnas, es decir si en el buscador pongo la "R", van a aparecer todas las filas donde haya una "R", si pongo "Gino", van a aparecer todas las filas que tengan un "Gino", etc.

A su vez hay 3 botones para aceptar al usuario preinscrito, editar y borrar.

EDITOR DE PREINSCRIPCIÓN

Editar pre-inscripción

Entregó certificado de salud?:

Entregó foto carnet?:

Entregó fotocopia del DNI?:

N° Preinscripción	Nombre	DNI	Dirección	Número de teléfono	Foto carnet	Certificado de salud	Fotocopia DNI	Acciones		
1	Gino Rossi	42133081	21:00:00 GMT-0300 (hora estándar de Argentina)	ginorossiar@gmail.com	B° Centenario Casa N° 209	No	No	No	No	<input type="button" value="Confirmar"/> <input type="button" value="Editar"/> <input type="button" value="Borrar"/>

Al presionar "editar" le permite al administrativo actualizar ese usuario, confirmar si entregó el certificado de salud , foto carnet ,fotocopia dni, etc. También es MODAL.

BUSQUEDA DE USUARIO

[INICIO](#)[CONOCENOS](#)[NOTICIAS Y EVENTOS](#)[MOODLE](#)[CONTACTO](#)[DESCONECTARME](#)

Buscar usuario

* La búsqueda puede ser tanto para el ID, como para el nombre de usuario, nombre completo o rol.

Buscar...			
ID	Nombre de usuario	Nombre completo	Rol
1	ndimmne0	Nappie Dimmne	Administrador
2	cteager1	Chrotoem Teager	Estudiante
3	hpattenden2	Hector Pattenden	Estudiante
4	apettyfar3	Angelita Pettyfar	Profesor
5	sdownie4	Selig Downie	Profesor
6	plouthe5	Paolo Louthe	Profesor
7	ibrakewell6	Iona Brakewell	Estudiante
8	cstump7	Clementia Stump	Estudiante
9	lbaldam8	Louie Baldam	Estudiante
10	pruf9	Pru Ruf	Estudiante

En la otra acción disponible del panel de control, el administrativo puede buscar usuarios, su ID, nombre completo y rol de hacer el FILTRADO. Si pone "administrador" solo aparece el administrador, si pongo "estudiante" solo los estudiantes, etc.

WEB ACTUAL ISFT 190

Limitaciones de Joomla

- Se pagan algunos plugins que añaden un coste adicional al desarrollo.
- Problemas de compatibilidad de los plugins que requieren programación PHP para resolver.
- El problema de los parámetros del servidor con los sitios grandes.
- Las opciones de ajuste para el usuario son limitadas, no hay facilidades avanzadas para el usuario.

Compartido con la página actual del ISFT, mostramos desventajas y las comparamos con el prototipo de la nueva página:

- **Inicio:**
 - Carrusel muy grande con carreras: Lo primero que se puede apreciar es la existencia de un carrusel excesivamente grande, el cual dificulta la apreciación de las demás secciones, que a nuestro entender deberían tener una ponderación más significativa.
 - La información relevante se encuentra a mitad de la página en un color poco amigable, con un hover con colores que puede dificultar la lectura de la información. Además la “previsualización” no es tal, no existe. Es la noticia completa en un primer plano. Al “ingresar a la noticia” solo nos redirige a una página el cual contiene EL MISMO CONTENIDO previamente “pre visualizado”. Por ende, la misma genera redundancia de la misma información, formularios de registros con vistas no muy agradables para el usuario.
- **Carreras:** Carreras no es muy claro a la hora de usar, interfaz poco amigable para el usuario, no intuitivo para el alumno (nadie)
- **Eventos:** No se entiende la interfaz, no es intuitiva, falta información en algunos apartados, y no funciona inclusive el botón.
- **Galería de Imágenes:** Debería estar dentro de eventos/noticias como un apartado, interfaz no intuitiva.
- **Contacto:** Interfaz poco intuitiva, no se entiende algunos apartados, el mapa de google no funciona.
- **Login:** Si bien puedes recuperar la contraseña (vía email) con código de verificación (tanto para el usuario como contraseña “eso sí separados en cada consulta”), al iniciar sesión en la página no tiene ninguna funcionalidad, ni para el alumno ni para el docente. No tiene ninguna característica nueva, ni ninguna función que amerite la registración a la página.

Casos de Uso

Usuario-Invitado:

- Solicitar información de la página (Lectura de la página)
- Solicitar una preInscripción.
- Compartir en redes sociales.
- Realizar consultas (Formulario)

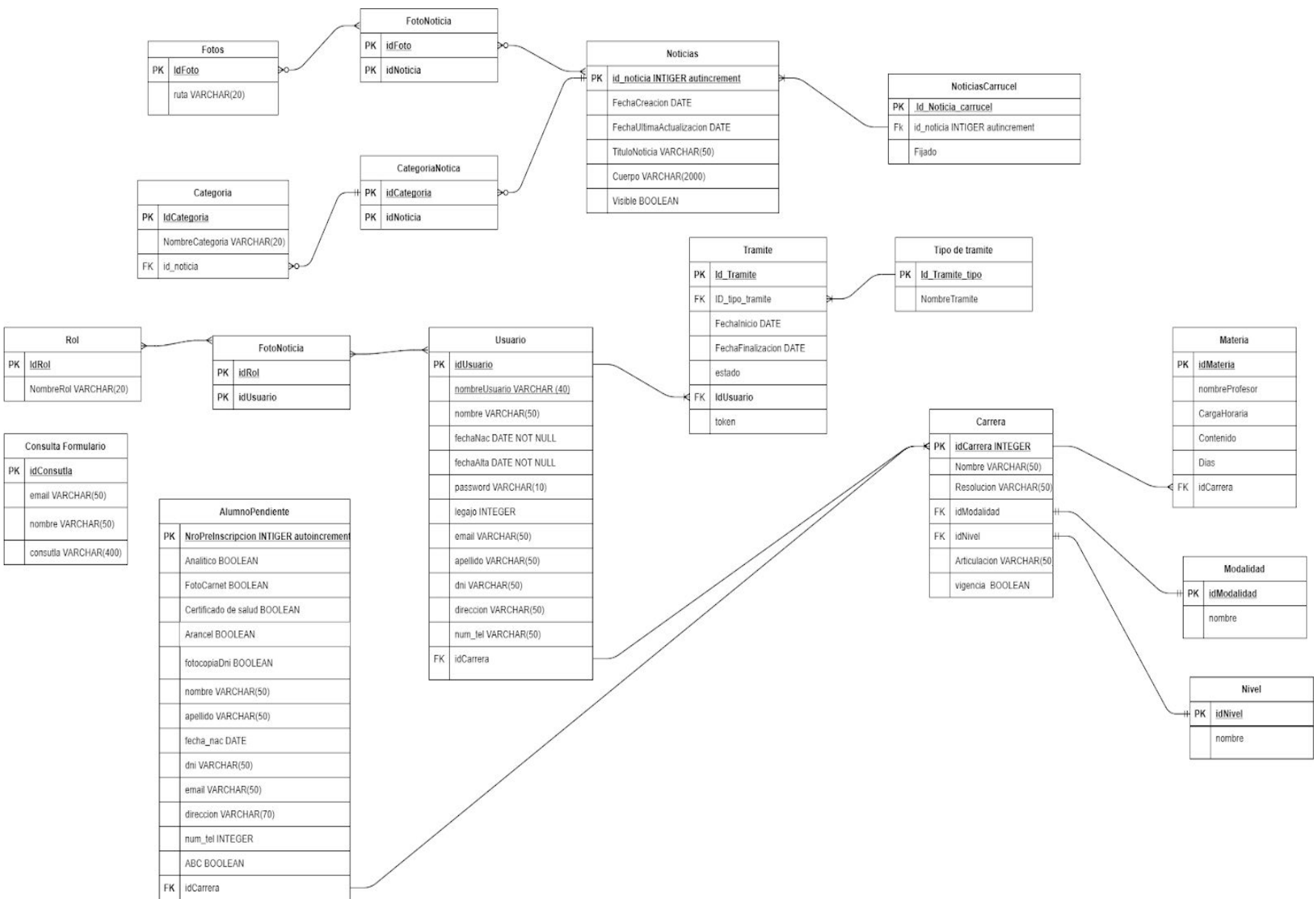
Alumno

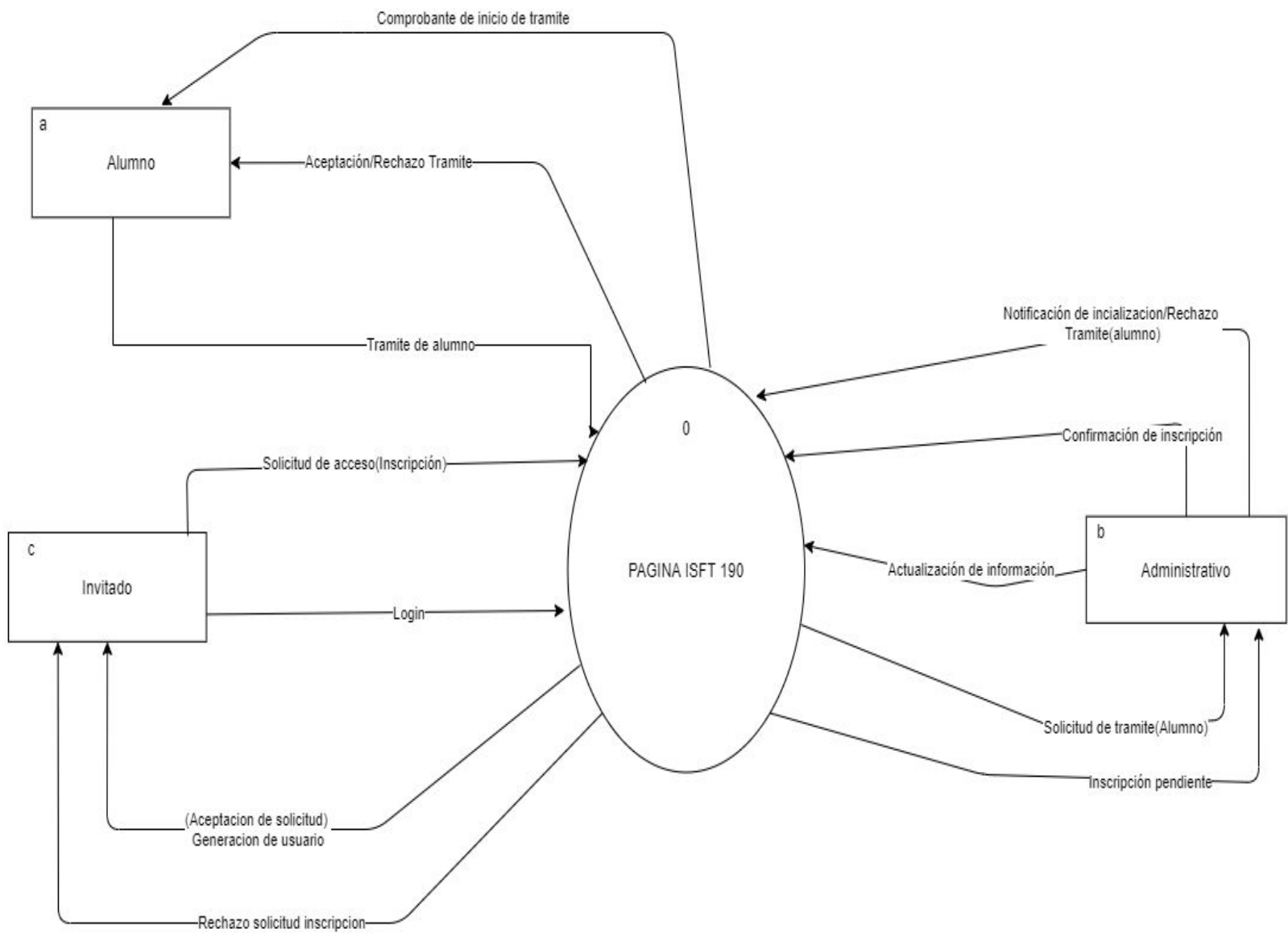
- Login en la página.
 - Solicitar trámite.
- Ingresar a moodle.
- Recuperar contraseña.
- Consultas de temas académicos al Instituto.

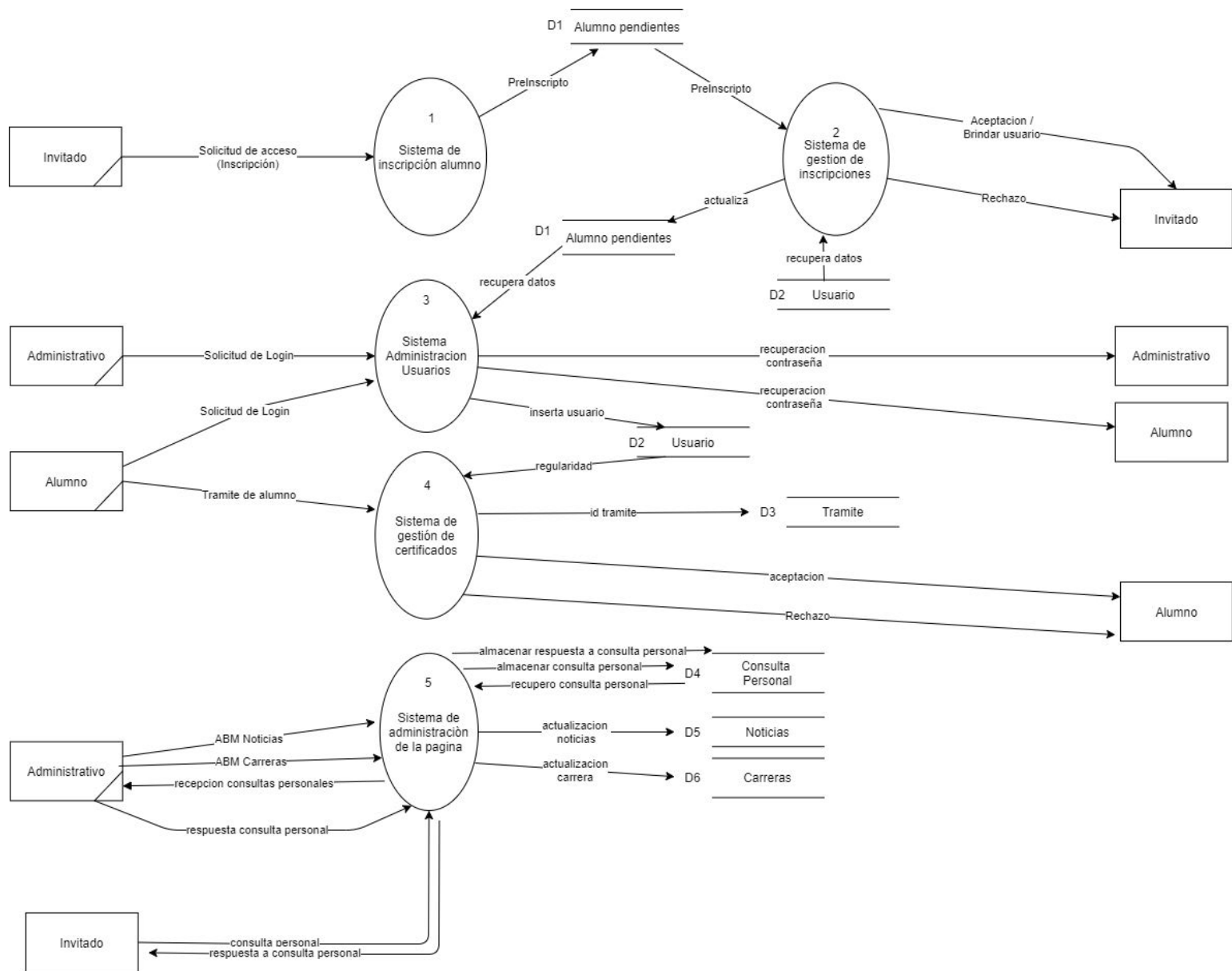
Administrativo

- Login Administrativo.
- ABM noticias.
- Gestión de trámites.
- Respuesta a consultas.
- Recuperar contraseña.

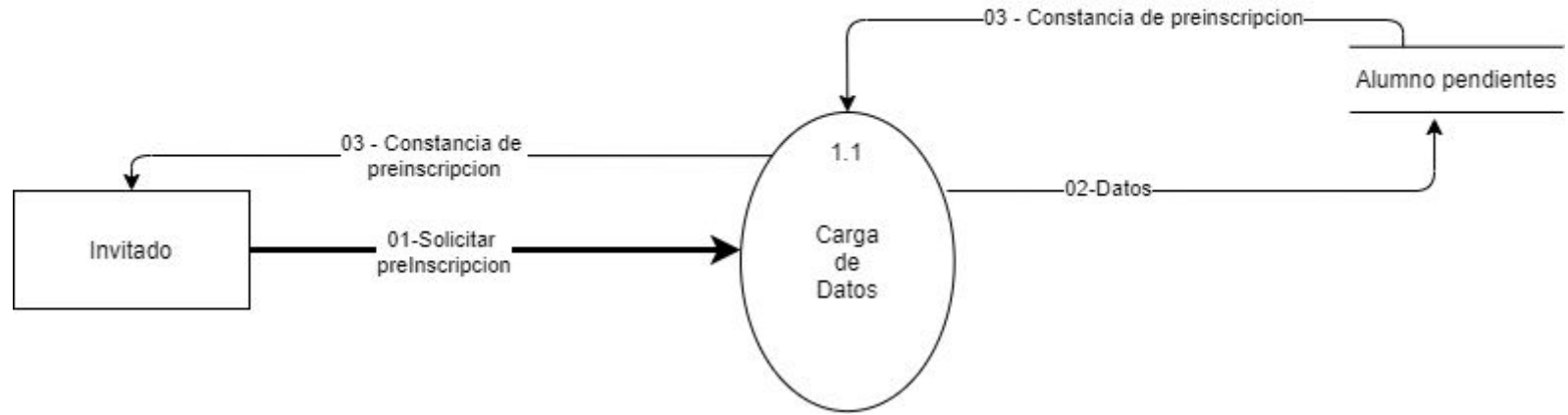
Diagramas



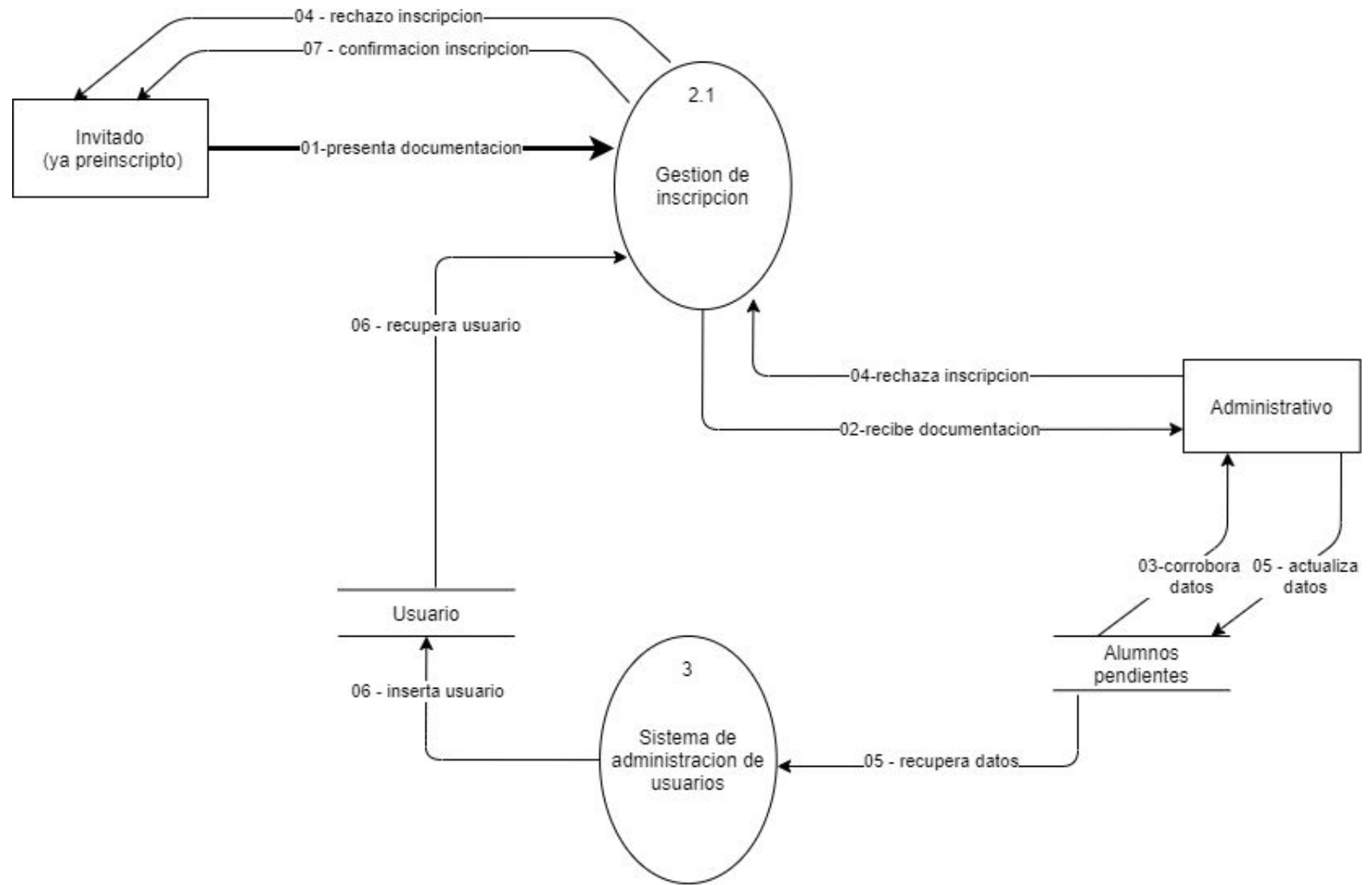




Sistema de inscripción de alumno

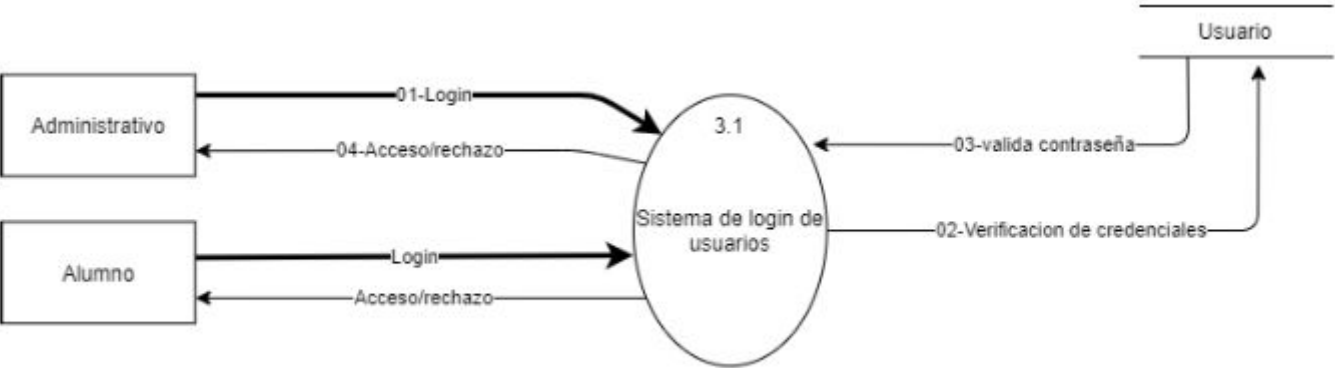


Sistema de Gestion de inscripcion

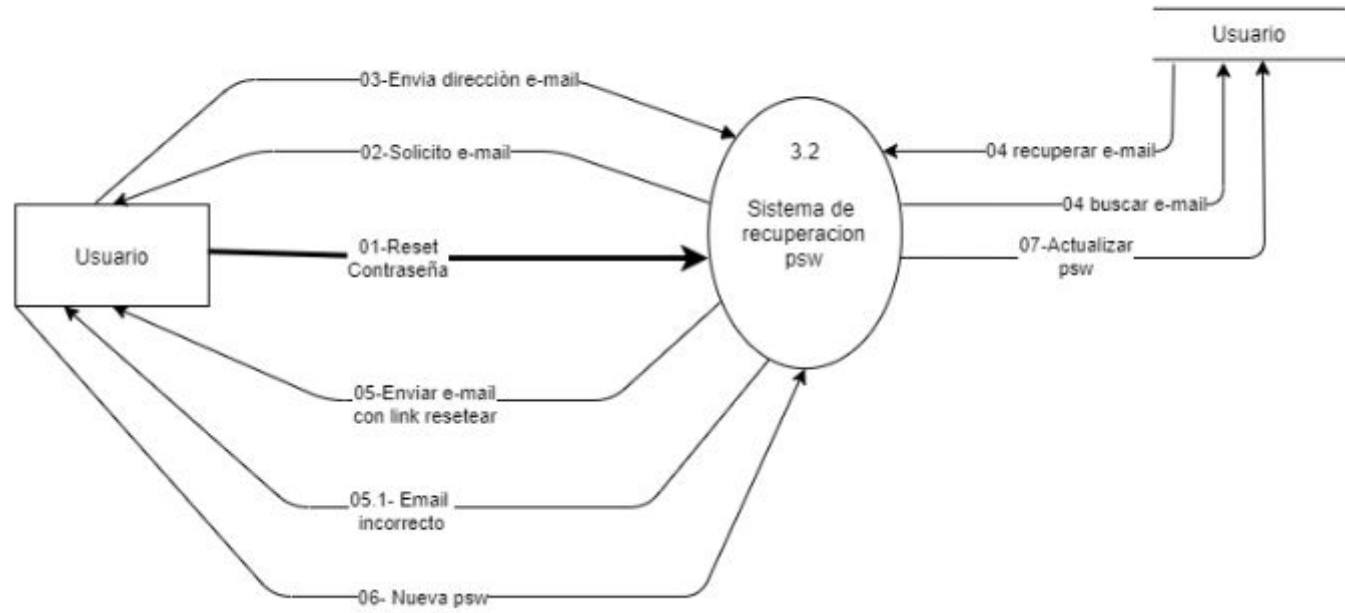


Sistema de administración de usuarios

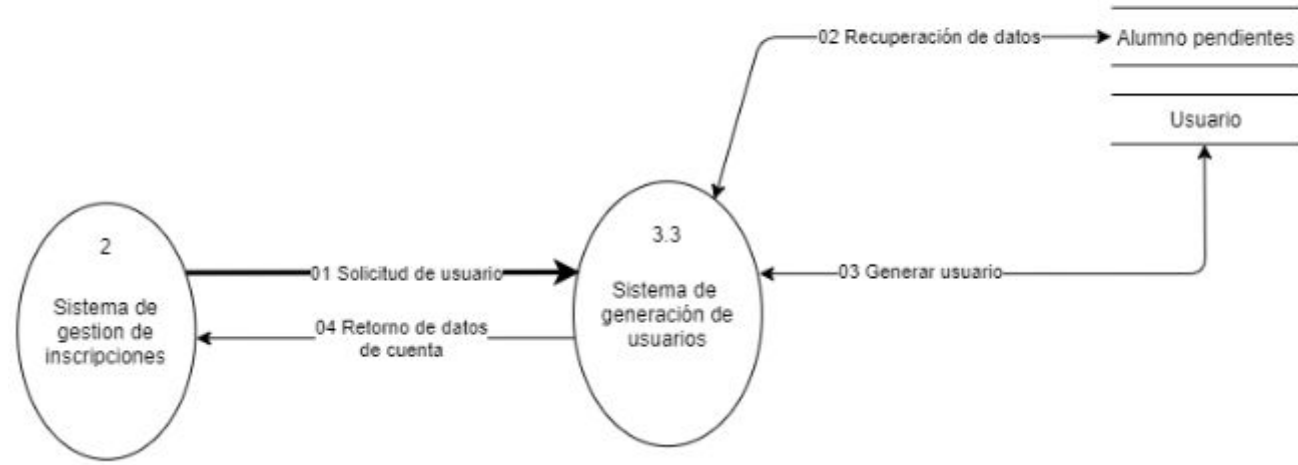
Sistema de login



Sistema de recuperar contraseña

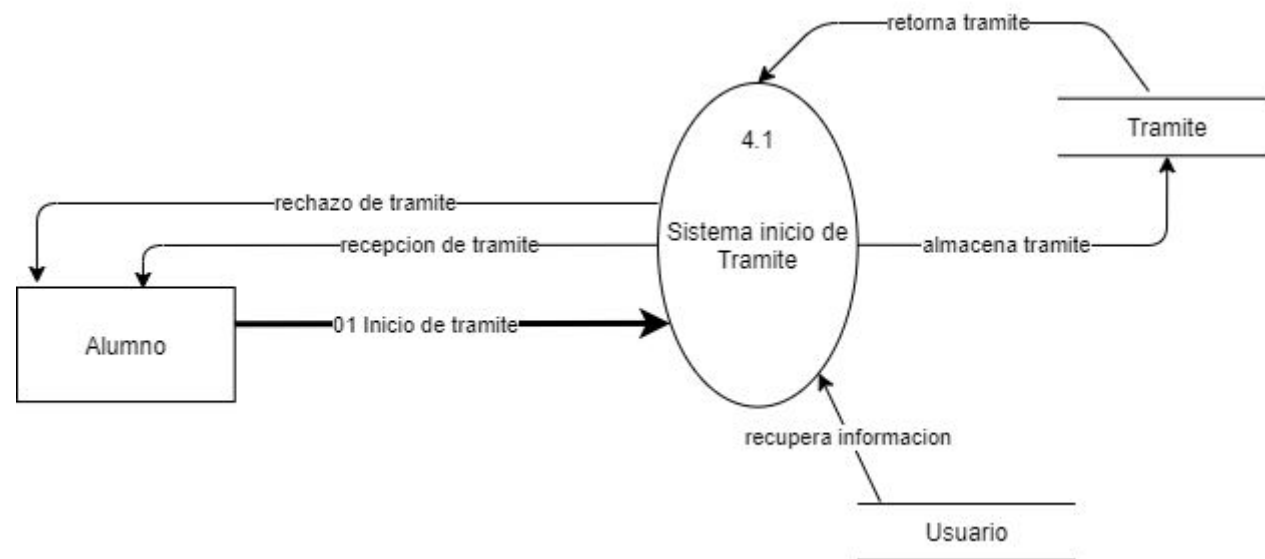


Sistema generación de usuarios

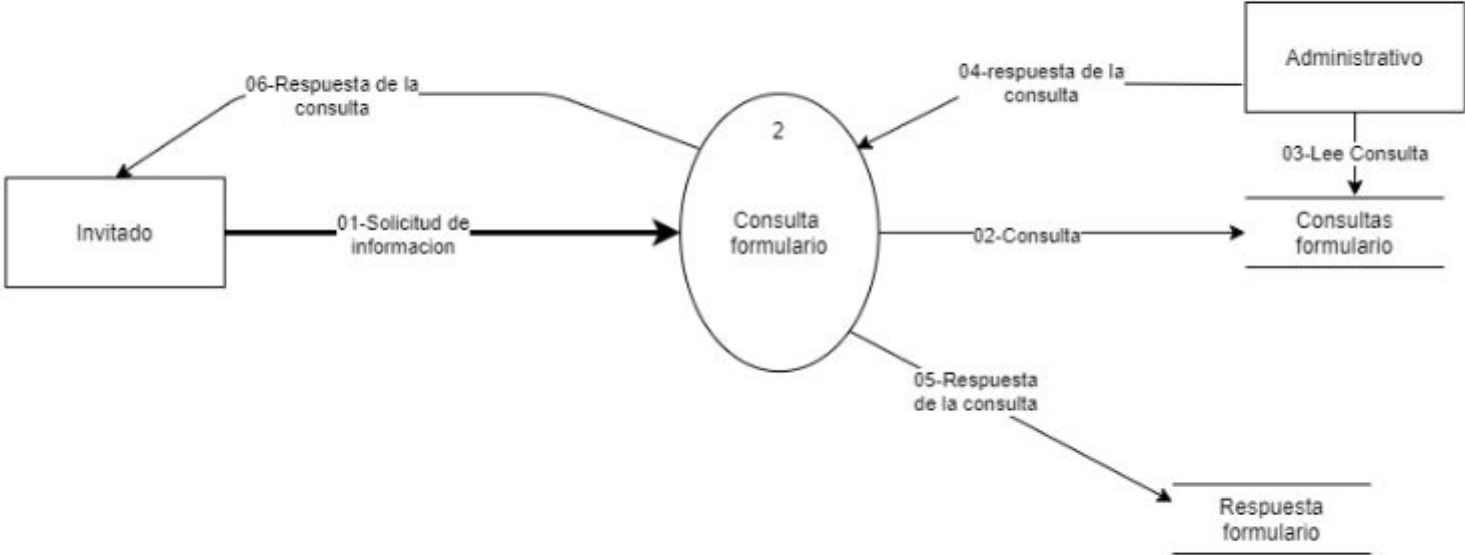


Sistema de gestión de tramites

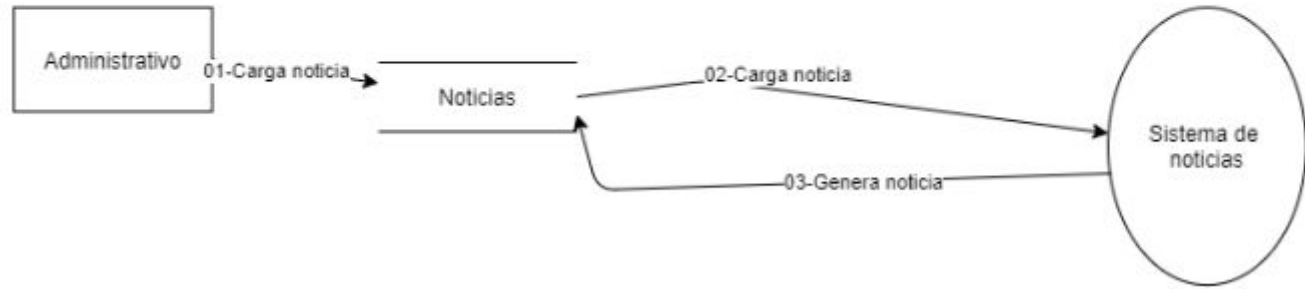
Sistema de solicitud de tramite



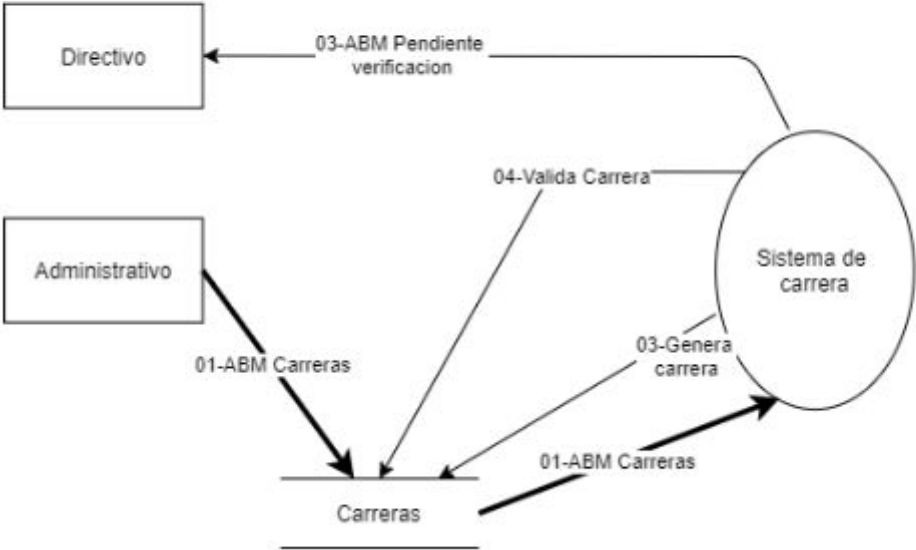
Sistema de consulta mediante formulario



Sistema de administración de noticias



Sistema de administración de carreras



Página Principal

Frontend

- Barra de navegación (fixed o estática)
 - Inicio
 - Instituto
 - Autoridades
 - Dirección
 - etc etc
 - Contacto
 - Carreras
 - Trabajos realizado en el instituto(?)
 - Novedades/noticias
 - Faq
 - Login/Logout
- Menú(asociado a la sesión)
 - Tramite de alumno
 - Certificado alumno regular
 - Inscripción a examen
 - Docentes
- Carrusel de noticias
- Sección
 - Alumnos
 - Docentes
 - Ingresantes
- Footer
 - Dirección
 - Mapa ([generador de iframes de google maps](#))
 - Redes sociales

BackEnd

- Instalar Express
- Base de datos
 - Conexión con la base de datos
 - Mysql/Postgres
 - Usuarios
 - Noticias
 - Sequelize
 - Migraciones
 - Variables de entorno
 - Datos de conexión hacia apis/bd/servicios externos
- Registrar usuarios nuevos(usar los de moodle?)
- Login/Logout usuarios
- Vue--"Conexión"
- Api
 - moodle
 - Bd del inst(Clarion)

BackEnd funcionalidades implementadas

1. Desplegar Información del Instituto

2. Login de usuario a nivel Administrativo

a. ABM--Usuarios

- i. /src/modelo/usuarios.js ✓
 - 1. operaciones de encriptar contraseñas ✓
 - 2. operación de desencriptar contraseñas ✓
- ii. /src/controladores/controladorUsuarios.js ✓
 - 1. crearUsuarios ✓
 - 2. encontrarUsuarios ✓
 - 3. actualizarUsuarios ✓
 - 4. borrarUsuarios ✓
 - 5. Verificación de permisos
- iii. /spec/Modelo/usuario.spec.js

b. ABM--Noticias

- i. /src/modelo/noticias.js ✓
- ii. /src/controladores/controladorNoticias.js ✓
 - 1. crearNoticia ✓
 - 2. encontrarNoticia(filtrado por fecha) ✓
 - 3. actualizarNoticia ✓
 - 4. borrarNoticia ✓
 - 5. validarNoticia ✓
 - 6. fijarNoticiaCarrucel ✓
 - 7. Verificación de permisos ✓
- iii. /spec/Modelo/usuario.spec.js

c. ABM -- Carrera ✓

- i. /src/modelo/carreras.js ✓
- ii. /src/controladores/controladorCarreras.js ✓
 - 1. crearCarrera ✓
 - 2. encontrarCarrera(filtrado por fecha)
 - 3. actualizarCarrera ✓
 - 4. borrarCarrera ✓
 - 5. validarCarrera
 - 6. Verificación de permisos ✓
- iii. /spec/Modelo/usuario.spec.js

d. ABM --InfoPagina ✓

- i. /src/modelo/informacionPagina ✓
- ii. /src/controladores/controladorInformacionPagina ✓
 - 1. Nombre del INST ✓
 - 2. Imagen encabezado ✓
 - 3. favicon (Icono desplegado en la pestaña) ✓

- 4. metaTag (FRONT)✓
- 5. Barra de navegación✓
 - a. inicio✓
 - b. carreras (Agregar Carreras)✓
 - c. infoIngresantes
 - d. Login✓
 - e. Moodle

3. Login de usuarios a nivel Alumno

- a. Solicitud Certificados.✓
 - i. alumno regular
 - ii. historial carrera
 - iii. analitico
 - iv. etc.
 - v. generación de comprobante✓

Conclusión

A vísperas de la finalización de este año atípico, en el cual nos vimos desafiados por la pandemia y todo aquello que arraigó a nuestro contexto académico (problemas de conectividad; organización, virtualidad y trabajo en equipo remoto, etcétera) la realización de este proyecto nos enfrentó a problemáticas que, las cuales nos resultaron extremadamente educativas para adentrarnos en el contexto de una aproximación a lo que podría ser nuestro futuro laboral como Analistas de Sistemas, empleando tecnologías modernas, herramientas de trabajo de control de versiones y trabajo cooperativo como lo es Git.

Si bien los plazos de realización del proyecto fueron largos, el hecho de tener que incursionar en tecnologías desconocidas para algunos de nosotros, nos implicó una mayor cuota de esfuerzo, al tener que aumentar la carga horaria de dedicación al proyecto. Esto considerando todo lo que conlleva la curva de aprendizaje de lenguajes de programación, las buenas prácticas, los sistemas de gestión de bases de datos y el anteriormente nombrado Git.

Bajo este contexto, se han cumplido con los objetivos de rediseñar, reestructurar y añadir nuevas funcionalidades a la página web del Instituto. En caso de considerarse apropiada la inversión de alojar la plataforma en un servidor propio del Instituto, se podría acceder a la totalidad de las funcionalidades. En esta oportunidad se trata de un prototipo experimental, es decir, es una versión acotada de lo que representa la versión final y completa.

En el transcurso de la realización de este proyecto nos hemos capacitado para trabajar eficientemente en grupo, desarrollando en conjunto los diagramas pertinentes al sistema, el informe y los requerimientos del mismo, dándonos como experiencia el poder compartir nuestras opiniones y decisiones, así como debatir las metodologías necesarias para implementar dichos elementos.

Teniendo como base el desarrollo de este sistema, se considera abierta y completamente factible la posibilidad de implementar aún más funcionalidades que aporten un mayor desempeño para la plataforma, siendo esta una buena oportunidad para que los alumnos que se encuentran en tercer año de Análisis de Sistemas puedan experimentar y trabajar a la par de tecnologías modernas y funcionales.

Glosario

Carreras

Nombre abreviado: Nombre más descriptivo de la carrera en cuestión. En nuestro caso, el nombre completo es “Tecnicatura Superior en Análisis de Sistemas” pero por comodidad y otras razones suele denominarse “Analista de Sistemas” o “Análisis de Sistemas”.

Modalidad: Especifica el tipo de cursado de la carrera (presencial o a distancia).

Tipo: Si es carrera vigente o a ciclo cerrado.

Duración: Duración de la carrera en años.

Horas: Cantidad de horas cátedra de la carrera.

Nivel: Especificación respecto a si es un curso, diplomatura o carrera terciaria.

Roles

Invitado: Es el usuario que visita la página, no tiene ningún rol en especial, es el usuario que nunca se inscribió en la página.

Alumno: Es el usuario que se inscribió en la página, y pertenece por ende al Instituto.

Administrador: Es el usuario que administra la página en su totalidad. Tanto las carreras, como las Noticias, la respuesta a los Usuarios, etc.

Nombre abreviado: Es la jerarquía que tendrá el Usuario en la página.

Diagramas

- Diagrama Entidad Relación(En progreso)
 - https://drive.google.com/file/d/1IEvuODVodFUWqdfkMFZ8mfJEHC75_geV/view?usp=sharing
 - APUNTE MÁS [DETALLADO DEL DFD](#)

Bibliografía

Documentación oficial de Bootstrap

- <https://getbootstrap.com/>

Documentación oficial de Slider Revolution

- <https://www.sliderrevolution.com/>

Documentación oficial de Node:

- <https://nodejs.org/es/docs/>

Documentación de las diferentes API:

- <https://nodejs.org/dist/latest-v14.x/docs/api/>

Documentación de Express ¿Cómo instalar express?

- <https://expressjs.com/es/starter/installing.html>

Documentación de Express, buenas prácticas de seguridad:

- <https://expressjs.com/es/advanced/best-practice-security.html>

Documentacion Postgresql

- <https://www.postgresql.org/docs/>

Express-session

- <https://www.youtube.com/watch?v=J1qXK66k1y4>

Git

- <https://chris.beams.io/posts/git-commit/>
- <https://www.freecodecamp.org/news/writing-good-commit-messages-a-practical-guide/>
- <https://www.youtube.com/watch?v=jSJ8xhKtfP4&list=PLTd5ehlj0goMCnj6V5NdzSIHBgrIXckGU>

JEST

- https://www.reddit.com/r/javascript/comments/7vy45n/why_would_you_choose_jasmine_over_jest/ <- Jasmine vs Jest
- <https://jestjs.io/docs/> <- Documentación JEST
- <https://www.youtube.com/watch?v=7r4xVDI2vho> <- Crash Course JEST (57 min)
- <https://jestjs.io/docs/en/architecture> <- Video (también 57min) arquitectura JEST

Manejo de roles

- <https://hisk.io/role-based-authentication-with-angular-express-jwt-mysql-part-1/>
- <https://hisk.io/role-based-authentication-with-angular-express-jwt-mysql-part-2/>

NodeJS

- <https://www.toptal.com/nodejs/por-que-demonios-usaria-node-js-un-tutorial-caso-por-caso>
- <https://sodocumentation.net/es/node-js/topic/2975/despliegue-de-aplicaciones-node-js-en-produccion>
- <https://www.toptal.com/nodejs/por-que-demonios-usaria-node-js-un-tutorial-caso-por-caso>

Node and Sequelize

- <https://grokonez.com/node-js/sequelize/sequelize-crud-mariadb-example-build-crud-node-js-express-restapis-example>

Object Relational Model

- <https://programarfacil.com/blog/que-es-un-orm/>

Postgresql

- <https://stackoverflow.com/questions/60951462/what-is-the-usage-of-a-nologin-user-in-the-postgresql>
- <https://www.youtube.com/watch?v=-2kYJ0gZmCo>
- <https://wakervall.wordpress.com/2018/02/03/configuracion-de-usuarios-postgresql/>

Sequelize

- <https://sequelize.org/master/manual/validations-and-constraints.html#per-attribute-validations>
- Sembrado de datos Falsos
 - <https://www.npmjs.com/package/faker>
 - <https://www.youtube.com/watch?v=o4IDSvesITg>
- <https://sequelize.org/master/manual/migrations.html#creating-the-first-model--and-migration->
- <https://medium.com/@andrewoons/how-to-define-sequelize-associations-using-migrations-de4333bf75a7>

Seguridad--Backend

- <https://owasp.org/www-project-top-ten/>
 - Usuario con mínimo privilegios
 - Firewall de api// APigateWay (Arquitectura de microservicios)

Variables de entorno(Estado de NODE_ENV)

- <https://dev.to/flippedcoding/difference-between-development-stage-and-production-d0p>