# UNIVERSIDAD DON BOSCO



# Desarrollo de Software para Móviles DSM104

**INTEGRANTES**

**Kevin Eliu Pleitez Hércules 01T/03L**

**Oswaldo Alberto Valle Serrano 01T/03L**

**Roberto Fernando Funes Morán 02T/02L**

**Proyecto: Clínicas Medicas**

**Fecha de Entrega: 25/04/2021**

INDICE

[Requerimientos Técnicos 3](#_Toc70191875)

[Requisitos de Hardware 3](#_Toc70191876)

[Requisitos de Software 3](#_Toc70191877)

[Herramientas utilizadas para el desarrollo 4](#_Toc70191878)

[1. React Native 4](#_Toc70191879)

[2. Strapi 4](#_Toc70191880)

[3. Mongo DB 4](#_Toc70191881)

[4. Firebase 4](#_Toc70191882)

[Instalación de React Native CLI 5](#_Toc70191883)

[Instalación de Node JS y Chocolatey 5](#_Toc70191884)

[Instalación de Android Studio 5](#_Toc70191885)

[Configurando las variables para React Native 7](#_Toc70191886)

[Añadir ubicación de platform-tools 8](#_Toc70191887)

[Instalación y Configuración de Strapi 8](#_Toc70191888)

[Módulo de inicio de sesión 18](#_Toc70191889)

[Módulo de Registrar 21](#_Toc70191890)

[Módulo de Creación de consultas ( Pacientes ) 23](#_Toc70191891)

[Módulo de aceptar consultas (Doctor) 26](#_Toc70191892)

[Módulo de próximas Citas (Doctor) 28](#_Toc70191893)

[Diccionario de la base de datos 29](#_Toc70191894)

[User 29](#_Toc70191895)

[Avatar 29](#_Toc70191896)

[Mensajes 29](#_Toc70191897)

[Rooms 30](#_Toc70191898)

[Consultas 30](#_Toc70191899)

# Requerimientos Técnicos

## Requisitos de Hardware

* 4 GB RAM mínimo, 8 GB RAM recomendado
* 2 GB de espacio mínimo en el disco duro, 4 GB Recomendado (500 MB para el IDE + 1.5 GB para el Android SDK y el emulador de Android)
* 1280 x 800 resolución mínima de pantalla
* Microsoft Windows 7/8/10 (32- o 64-bit). El emulador de Android solo es soportado por versiones de 64 bits.

## Requisitos de Software

* Node JS
* Android Studio
* Java SE Development Kit 8

# Herramientas utilizadas para el desarrollo

## React Native

React Native es un framework JavaScript para crear aplicaciones reales nativas para iOS y Android, basado en la librearía de JavaScript React para la creación de componentes visuales, cambiando el propósito de estos para, en lugar de ser ejecutados en navegador, correr directamente sobre las plataformas móviles nativas, en este caso iOS y Android. Es decir, en lugar de desarrollar una aplicación web híbrida o en HTML5, lo que obtienes al final como resultado es una aplicación real nativa, indistinguible de la que podrías desarrollar con tu código en Objective-C o Java.

Esa es la teoría, pero veamos cómo propone React Native alcanzar este objetivo.

React Native usa el mismo paradigma fundamental de construcción de bloques de UI (componentes visuales con los que interacciona el usuario) que las aplicaciones nativas reales de Android e iOS, pero gestiona la interacción entre los mismos utilizando las capacidades de JavaScript y React.

## Strapi

Strapi es un CMS sin encabezado de código abierto basado en Node.js para todas las necesidades de API y gestión de contenidos de los desarrolladores.  
Permite crear API funcionales en cuestión de minutos y desarrollar software sin la molestia de las complejidades de un CMS. Todos los datos están disponibles a través de una API 100 % personalizable para adaptarse a cualquier necesidad. Se encarga del backend para que tú no tengas que hacerlo.

## Mongo DB

MongoDB es una base de datos distribuida, basada en documentos y de uso general que ha sido diseñada para desarrolladores de aplicaciones modernas y para la era de la nube.

## Firebase

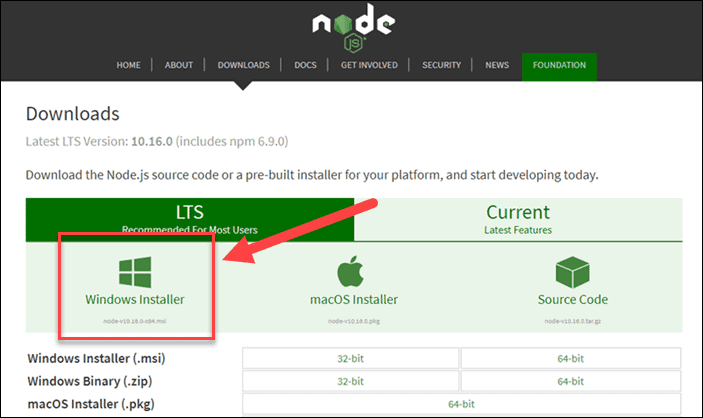
Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por Google en 2014.

Es una plataforma ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios y dando resultado también a la obtención de una mayor monetización.

# Instalación de React Native CLI

## Instalación de Node JS y Chocolatey

Paso 1: Descargar el instalador Node JS desde su sitio web oficial <https://nodejs.org/en/download/>. Y seguir los pasos internos del instalador



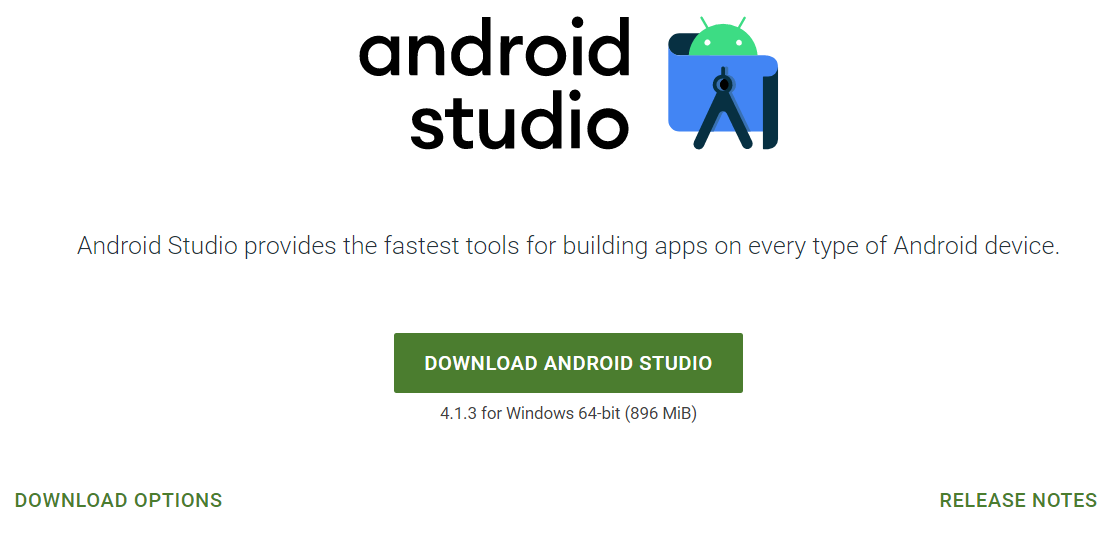
Paso 2: Una vez instalado Node JS se deberá ejecuta el siguiente comando para poder instalar “Java SE Development JDK 8”

* *choco install -y nodejs.install openjdk8*

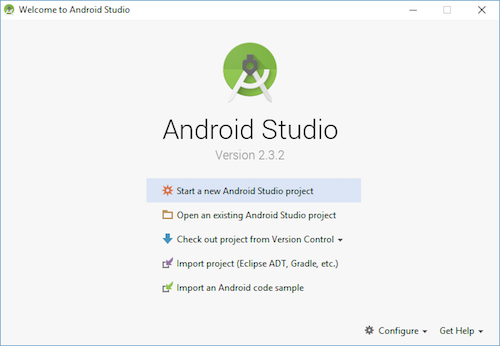
# Instalación de Android Studio

Paso 1: Descargar e instalar Android Studio desde su sitio web oficial y su usar el instalador siguiendo los pasos internos

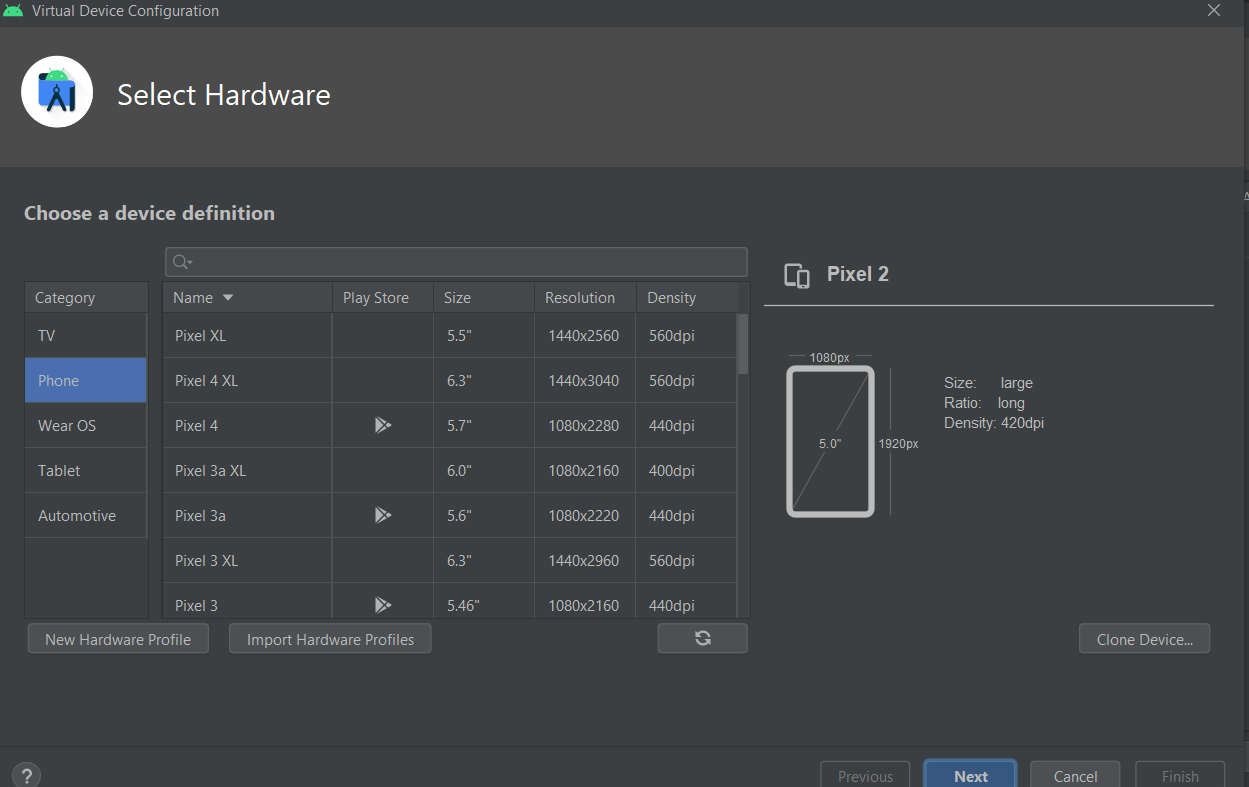
https://developer.android.com/studio



Paso 2: Una vez instalado deberemos configurar un celular para el emulador, dando clic en “*Configure*” luego en “*SDK Manager*” para elegir que versión de Android usaremos. Se recomienda Android 6.0 como mínimo.



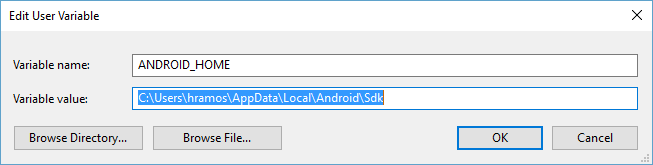
Paso 3: Abriendo Android Studio nos dirigiremos a “Tools” luego a “*AVD Manager*”, dentro dando clic en “*Create Virtual Device*” y elegiremos que celular queremos utilizar y la versión de Android



# Configurando las variables para React Native

Las herramientas de React Native requieren configuración de variables de entorno para el compilado de las aplicaciones.

1. Abrir **Panel de Control de Windows.**
2. Clic en **Cuentas de Usuarios,** entonces click **Cuentas de Usuaruis** devuelta
3. Clic en **Cambiar variables de entorno**
4. Clic en **Nuevo...**  para crear la variable de  ANDROID\_HOME que localiza a Android SDK



El SDK de Android se encuentra por defecto en la siguiente ubicación %LOCALAPPDATA%\Android\Sdk

Para añadir ANDROID\_HOME se puede realizar por medio del siguiente comando **Get-ChildItem -Path Env:\** en powershell

# Añadir ubicación de platform-tools

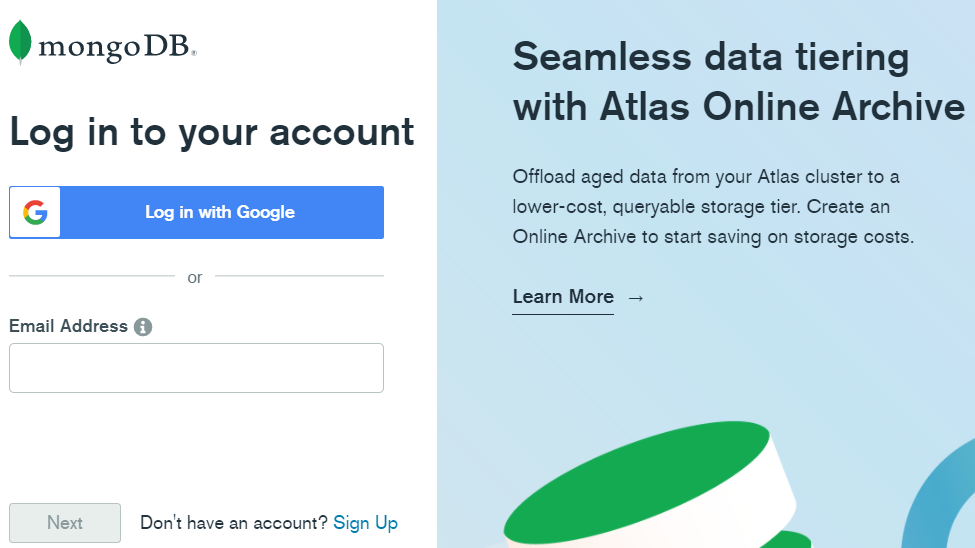
1. Abrir **Panel de Control de Windows.**
2. Clic en **Cuentas de Usuarios,** entonces clic **Cuentas de Usuaruis** devuelta
3. Clic en **Cambiar variables de entorno**
4. Seleccionar “Path”
5. Clic en Editar
6. Clic en **Nuevo** y añadir la platform-tools a la lista

La Platform-tools se encuentra por defecto en la siguiente ubicación: %LOCALAPPDATA%\Android\Sdk\platform-tools

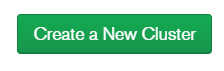
# Instalación y Configuración de Strapi

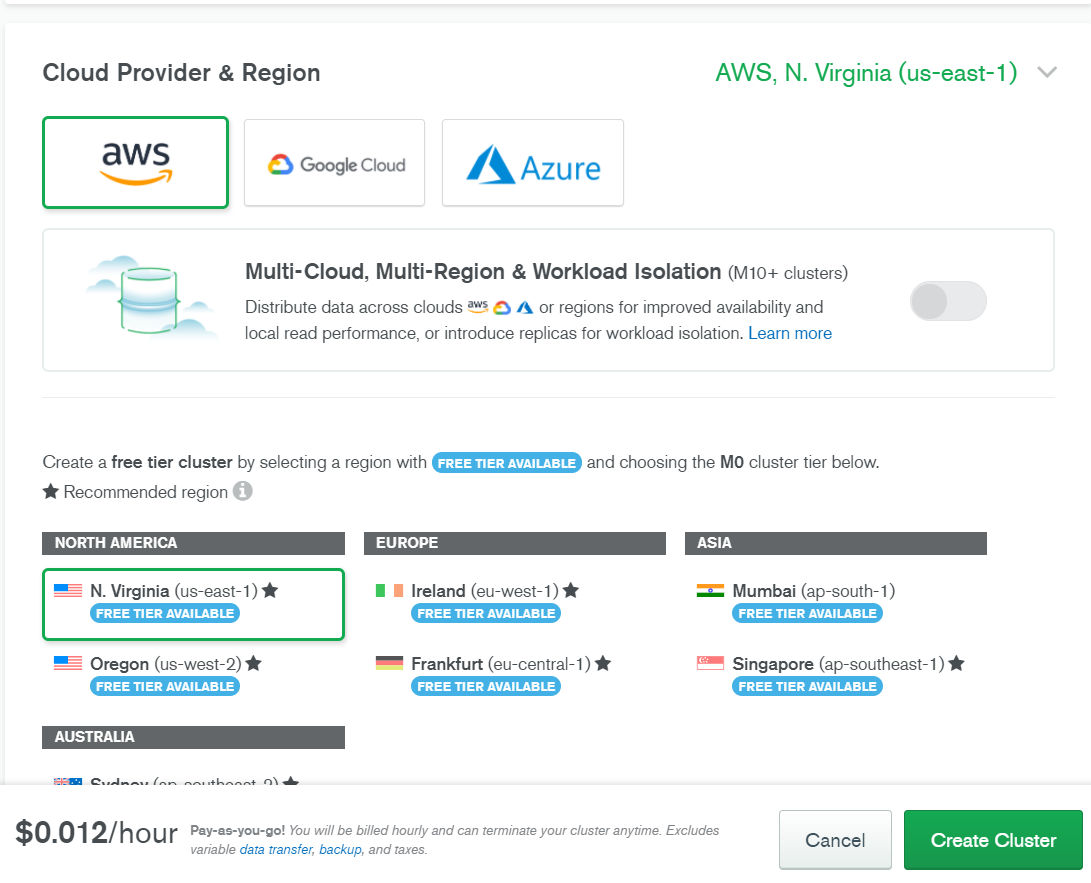
Paso 1: Debemos crear nuestra base de datos en Mongo DB, aunque para strapi podemos configurar para otra base de datos si deseamos como MySQL, Oracle o Postgresql. Ver documentación: <https://strapi.io/documentation/developer-docs/latest/setup-deployment-guides/configurations.html#database>

Para Mongo DB podemos tener nuestra base de datos en la nube de forma gratuita para cuando la aplicación se encuentra en el estado de desarrollo (development)

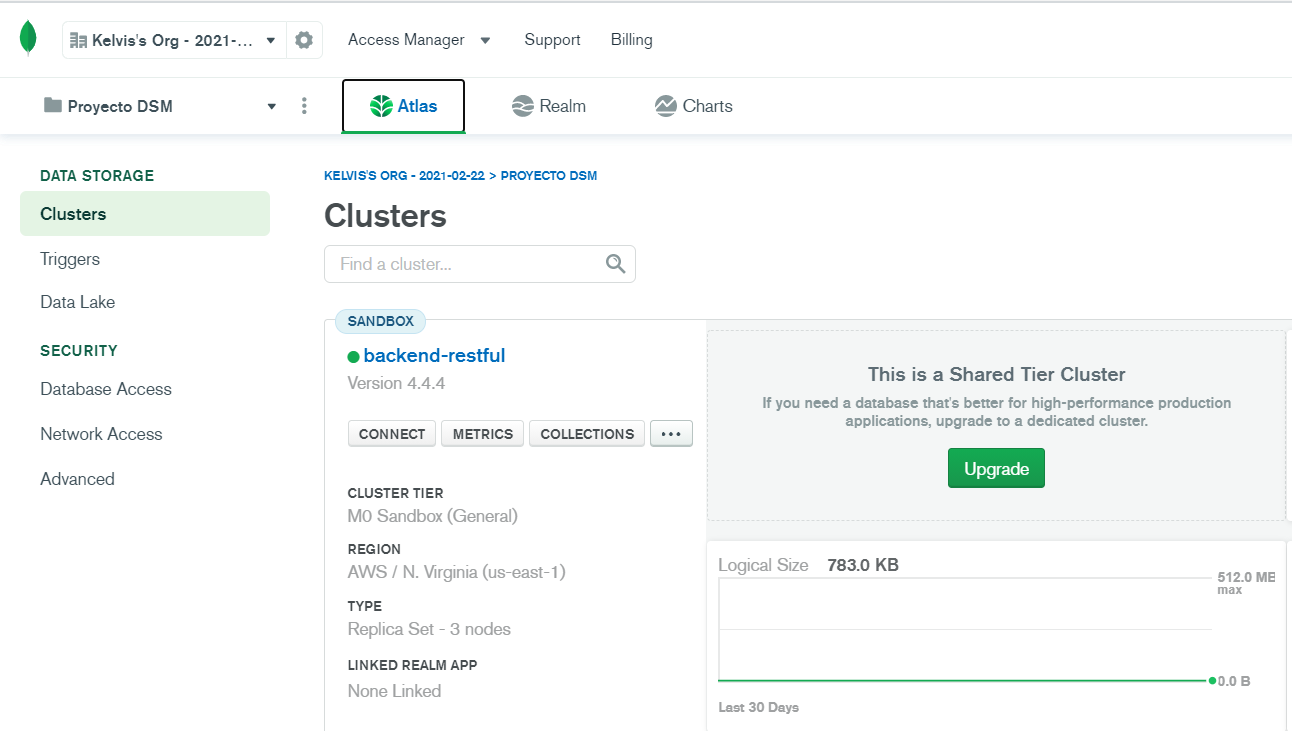


Paso 2: Después de entrar creamos nuestro primer Clúster, para nuestro primer clúster será gratuito

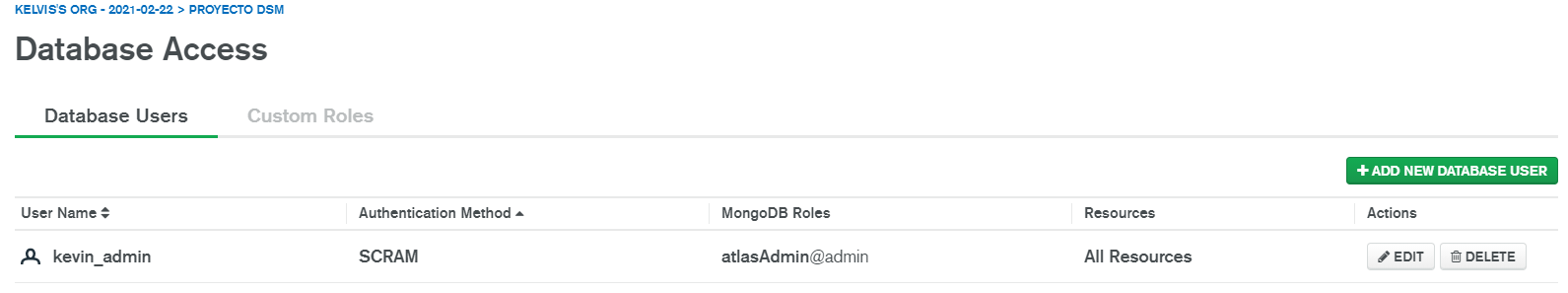


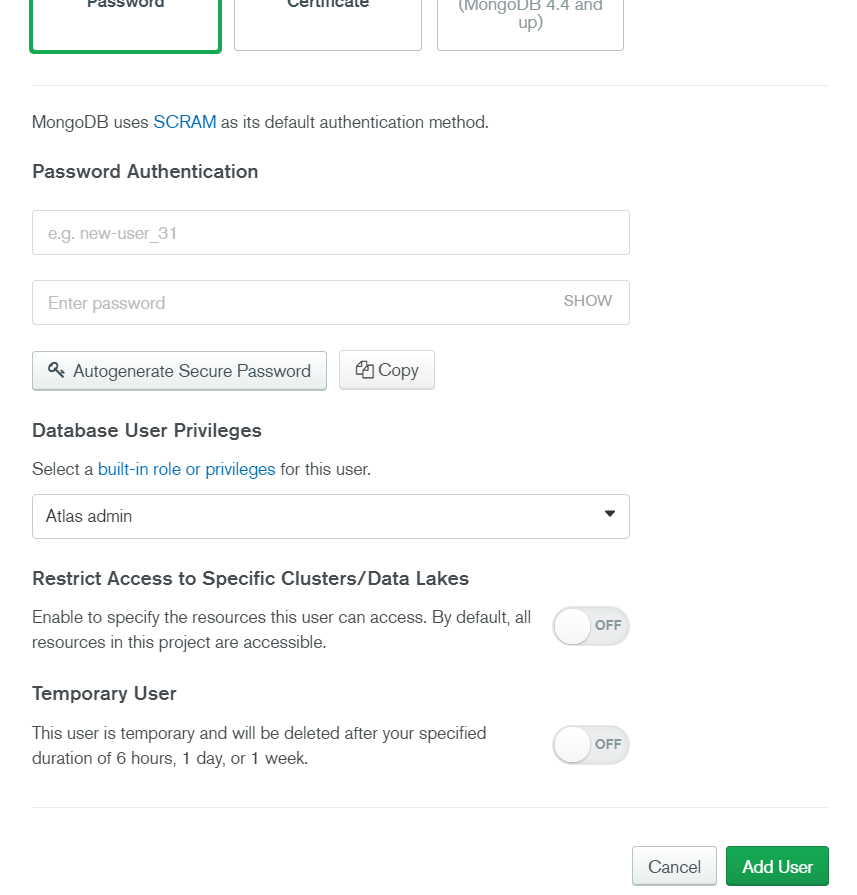
Paso 3: Para nuestro primer clúster será gratuito y tendremos hasta 500 horas por mes para poder utilizarlos de forma gratuita, si deseamos otro clúster o nos pasamos por las 500 horas este tendrá un cobro. Podemos elegir que servidor deseamos si AWS, Google Cloud o Microsoft Azure y su ubicación.  


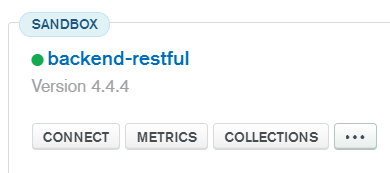
Paso 4: Cuando creamos el Cluster deberemos crear un usuario en “*Database Access*” para darle acceso a un usuario por el cual nos haremos inicio de sesión a través de strapi

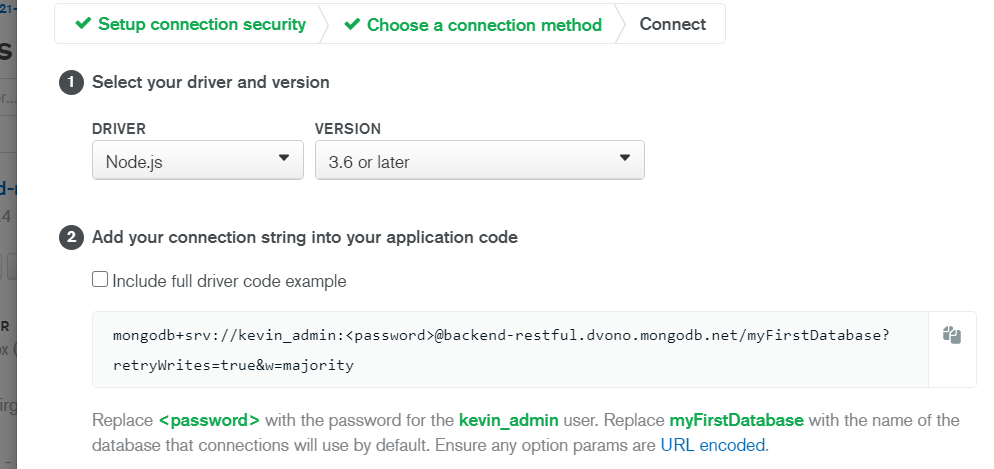


Paso 5: Damos clic en “*Add New Database User*” y dentro le daremos el rol de “*Atlas Admin*” para tener acceso total al clúster



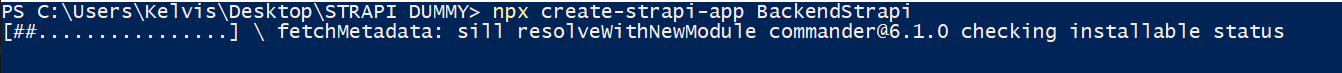


Paso 6: Para poder conectarnos a la base de datos damos click en “*connect*”, luego damos clic en “*Connect your application*”, seleccionamos “*Node JS*” para la versión más reciente y copiaremos la cadena de texto que nos presenta  


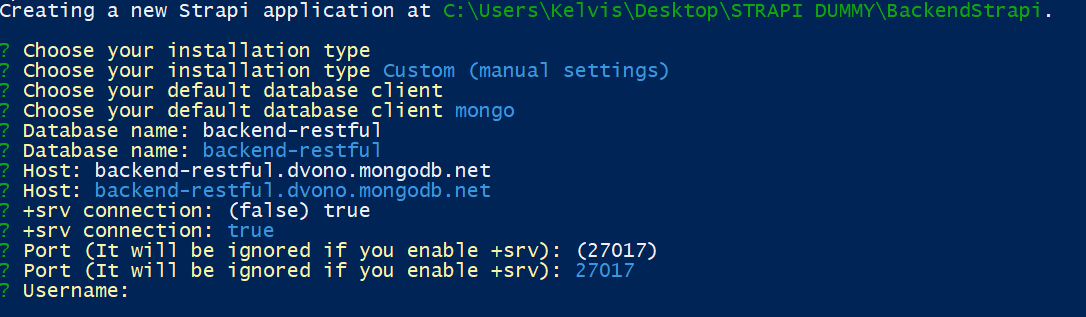


Nos interesa el host que sería en este caso: backendrestful.dvono.mongodb.net

Paso 7: Ahora que tenemos Mongo DB configurado deberemos crear nuestro servidor que será Strapi y ejecutamos el comando “*npx create-strapi-app NombreServidor* “y esperamos que descargue las dependencias.



Paso 8: Cuando se nos presenta la pantalla si queremos una instalación rápida o manual seleccionamos la instalación manual (Custom manual settings) y es importante que llenemos correctamente los campos el nombre de Database Name sea igual al nombre de nuestro cluster creado previamente, y en host será el que obtuvimos igualmente de mongo db. Una vez configurado esperamos que termine la instalación de Strapi.



Paso 9: Para nuestro backend Strapi utilizaremos distintos Providers que los locales para nuestro proyecto así mantendremos todo lo mas posible en la nube y evitar el tráfico para nuestro hosting. Añadiremos el Provider Google Firestorage y Provider Google login. Para nuestras las imágenes serán almacenadas en Google y debemos instalar el siguiente plugin ubicado en: <https://github.com/Lith/strapi-provider-upload-google-cloud-storage> y seguir las instrucciones dentro de este.

Paso 10: Para nuestro Provider Google debemos copiar el archivo original ubicado en *node\_modules/strapi-plugin-users-permission/services/Providers.js* y añadir las siguientes líneas de código al archivo y luego copiar el archivo modificado a la siguiente ubicacion */users-permissions/services/Provider.js*

    case 'google': {

*const* google = purest({ provider: 'google', config: purestConfig });

      google

        .query('oauth')

        .get('tokeninfo')

        // this is the only difference from the default google provider from strapi,

        // we'll be passing id\_token rather than access\_token similar to what's described here:

        // https://developers.google.com/identity/sign-in/android/backend-auth#using-a-google-api-client-library

        .qs({ id\_token: access\_token })

        .request((*err*, *res*, *body*) *=>* {

          if (*err*) {

            console.log(*err*)

            callback(*err*);

          } else {

            console.log(*body*)

            callback(null, {

              username: *body*.email.split('@')[0],

              email: *body*.email,

            });

          }

        });

      break;

    }

Paso 11: Con todo instalado ya podremos abrir el servidor y modificarlo para nuestro proyecto con el siguiente comando dentro de la raíz del proyecto “*yarn strapi develop”* y podremos gestionar el proyecto a través de la siguiente url <http://localhost:1337/admin>

Paso 12: Dentro del panel de control de Strapi nos dirigimos a “Creador de Tipos de Contenido”, luego clic en “Crear nuevo Tipo de Colección” y definiremos los campos que necesitaremos para nuestra base de datos.

Tabla 1 – Creación de registro de usuario local

|  |
| --- |
| Nombre: CU1 creación de Usuarios |
| Actor(es): Doctor/Paciente |
| Descripción: Se crean cuentas de usuario para ingreso al sistema por su respectivo rol, directamente en la tabla usuarios (user) |
| Flujo Normal:   1. El actor ingresa al módulo de registro 2. En la tabla usuarios (user) crea un nuevo registros con los datos del usuario (identificación, nombre completo, rol, email, username,provider,password,estado). 3. Se le asigna el rol de “Authenticated” y se confirma la cuenta |
| Flujo Alternativo: No aplica |
| Pos condición:   1. Cuenta creada |

Tabla 2 – Creación/Inicio de sesión de registro de usuario – Provider Google

|  |
| --- |
| Nombre: CU2 creación de Usuarios |
| Actor(es): Paciente |
| Descripción: Se crean cuentas de usuario para ingreso al sistema por su respectivo rol, directamente en la tabla usuarios (user) |
| Flujo Normal:   1. El actor ingresa al módulo de inicio de sesión 2. El actor accede a través del servicio de Google Play Services y selecciona una cuenta de Google 3. Se envía una petición a Google y se recibe un token por parte de este. 4. Se envía el token a Strapi 5. En la tabla usuarios (user) crea un nuevo registros con los datos del usuario (identificación, nombre completo, rol, email, username,provider,password,estado). 6. Se le asigna el rol de “Authenticated” y se confirma la cuenta |
| Flujo Alternativo: No aplica |
| Pos condición:   1. Cuenta creada 2. Se obtiene un token JWT por parte de strapi. 3. Se accede a la aplicación. |

Tabla 3 – Inicio de sesión de usuario local

|  |
| --- |
| Nombre: CU3 Inicio de sesión de Usuarios |
| Actor(es): Doctor/Paciente |
| Descripción: Acceder a la aplicación por medio de un inicio de sesión local (correo electrónico y contraseña) |
| Flujo Normal:   1. El actor ingresa al módulo de inicio de sesión 2. El actor ingresa sus datos personales como correo electrónico y contraseña. 3. En la tabla usuarios (user) se busca al usuario con respecto al correo electrónico y la contraseña correcta 4. Strapi regresa información del usuario que coincida junto a un token JWT que permitirá el acceso a la aplicación y dará permisos necesarios para realizar acciones dentro de la aplicación 5. Con la información retornada del servidor se almacena dentro de la aplicación (Frontend) de manera temporal con *Secure Storage* y así no perderla para mantener abierta la sesión. 6. Una vez la información ha sido almacenada se abre la aplicación en la pantalla principal que dependerá del rol tendrá variaciones. |
| Flujo Alternativo:   1. Si el usuario es Paciente su pantalla principal será las consultas creadas con anterioridad 2. Si el usuario es Doctor su pantalla principal será las consultas creadas por los Pacientes y cuyo estado sea falso. |
| Pos condición:   1. Información almacenada en la aplicación |

Tabla 4 – Registrar consulta

|  |
| --- |
| Nombre: CU4 creación de consultas |
| Actor(es): Paciente |
| Descripción: Se crea un registro de consulta en la tabla de “consultas” donde estará la información respectiva del caso con la información del Paciente y la espera de un Doctor que acepte el caso. |
| Flujo Normal:   1. El actor ingresa al módulo de crear consulta 2. En la tabla consultas se crea un nuevo registro con los datos del usuario, identificación, fecha cita, descripción, horas, estado. 3. Se asigna de manera predeterminada el estado de “falso” |
| Flujo Alternativo:   1. Una vez publicado la consulta se esperará si es aceptado el caso, en caso no haya modificaciones el estado permanecerá en “falso” 2. Si el caso es aceptado por un Doctor se registrará el usuario Doctor en la consulta |
| Pos condición:   1. Consulta creada. |

Tabla 5 – Modificación de usuario.

|  |
| --- |
| Nombre: CU5 modificación de Usuarios |
| Actor(es): Doctor/Paciente |
| Descripción: Modificar la información del usuario (user) y actualizar el perfil junto a la información almacenada. |
| Flujo Normal:   1. El actor ingresa al módulo de registro y modifica los datos almacenados que se imprimen 2. En la tabla usuarios (user) se actualiza el registro con la nueva información almacenada 3. Servidor strapi regresa la información actualizada |
| Flujo Alternativo: No aplica |
| Pos condición:   1. Información del perfil actualizado |

Tabla 6 – Eliminar consulta

|  |
| --- |
| Nombre: CU6 eliminación de consulta |
| Actor(es): Paciente |
| Descripción: eliminación de registros de consultas por parte del paciente |
| Flujo Normal:   1. El actor ingresa al módulo de consultas 2. El actor mantiene presionado la consulta que desea eliminar 3. Se envía la petición de eliminar hacia el servidor 4. Se busca la consulta con respecto a la identificación de la consulta y es eliminada de la base de datos. |
| Flujo Alternativo: No aplica |
| Pos condición:   1. Registro de consulta eliminado |

Tabla 7 – creación de Room

|  |
| --- |
| Nombre: CU7 creación de room |
| Actor(es): Paciente/Doctor |
| Descripción: creación de “room” para dos usuarios que desean establecer comunicación entre ellos pueda enviarse mensajes |
| Flujo Normal:   1. El actor ingresa al módulo de Chat 2. El módulo envía información del actor hacia el servidor 3. Se crea un room con la información de los actores que se desea establecer comunicación 4. Con el room creado se establece la línea de mensajes para los actores |
| Flujo Alternativo: No aplica |
| Pos condición:   1. Canal de comunicación para los actores establecido. |

Tabla 8 – registrar mensaje

|  |
| --- |
| Nombre: CU8 creación de mensaje |
| Actor(es): Paciente/Doctor |
| Descripción: Registro de mensaje |
| Flujo Normal:   1. Actor ingresa contenido del mensaje y lo envía 2. El mensaje enviado por el actor se registra los datos del actor y hacia quien va dirigido 3. Se registra el mensaje enviado dentro del servidor 4. Se envía una respuesta del servidor sobre el estado del mensaje |
| Flujo Alternativo: No aplica |
| Pos condición:   1. Mensaje registrado en la base de datos. |

Tabla 9 – eliminar mensaje

|  |
| --- |
| Nombre: CU8 eliminar mensaje |
| Actor(es): Paciente/Doctor |
| Descripción: Eliminar el mensaje por parte del actor y desechar el registro del mensaje en la base de datos. |
| Flujo Normal:   1. Actor selecciona el mensaje que desea eliminar 2. El mensaje enviado por el actor se elimina de la base de datos. 3. Se envía una respuesta del servidor sobre el estado del mensaje |
| Flujo Alternativo: No aplica |
| Pos condición:   1. Mensaje eliminado en la base de datos. |

## Módulo de inicio de sesión

Este módulo fue creado con el fin de poder acceder dentro de la aplicación y utilizarla además de también poder registrar usuarios e iniciar sesión con dicha cuenta.

A continuación, describimos el procedimiento:

1. Se ingresa al módulo de “Inicio de sesión”

Interfaz de usuario gráfica

Descripción generada automáticamente

1. Dentro de la pantalla se tienen dos metodos para poder acceder a la aplicación a traves de un metodo local por correo y contraseña

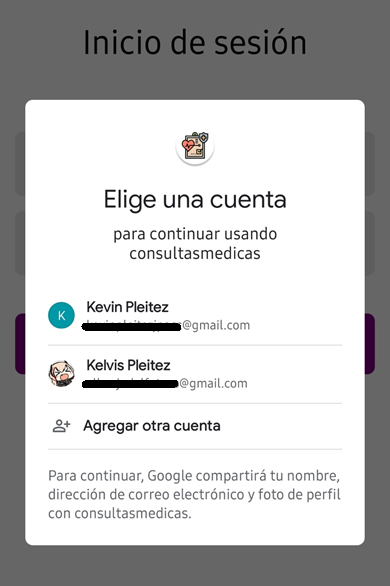
Interfaz de usuario gráfica

Descripción generada automáticamente

1. El otro metodo es accediendo a traves del botón de “Acceder con Google” utilizando cuentas de Gmail.

Interfaz de usuario gráfica

Descripción generada automáticamente



## Módulo de Registrar

Este módulo fue creado con el fin de poder crear una cuenta nueva dentro de la aplicación.

A continuación, describimos el procedimiento:

1. Accediendo desde la pantalla que se muestra en pantalla en el módulo de “Inicio de sesión”. A través del enlace “¿NO TIENES CUENTA? CREA UNA AHORA”

Interfaz de usuario gráfica

Descripción generada automáticamente

1. De forma inmedia en una nueva ventana se mostrara los campos requeridos para el formulario

Interfaz de usuario gráfica

Descripción generada automáticamente

1. En la opcion de “Selección su Rol” podremos elegir que rol deseamos crear nuestra cuenta, posteriormente se hara una revision interna para los que desean el rol de “Doctor”.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Uno de los campos que necesitamos llenar es el campo de imágenes que se nos mostrara a traves de un boton para poder seleccionar una imagen de nuestra galería

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Una vez llenado los campos enviamos la información del formulario al servidor de Strapi para poder registrar al nuevo usuario dentro del sistema.

Interfaz de usuario gráfica

Descripción generada automáticamente

1. Una vez creada la cuenta con éxito se nos regresara de forma inmediata a la pantalla principal para poder iniciar sesión con nuestros datos personales.

## Módulo de Creación de consultas ( Pacientes )

Este módulo fue creado con el fin de poder crear una cuenta consultas en los registros dentro de la aplicación.

A continuación, describimos el procedimiento:

1. A través del menú de navegación dentro de la aplicación el usuario de paciente puede acceder a esta pantalla donde tendra los siguientes elementos requeridos para la realizacion de la consulta

Captura de pantalla de un celular

Descripción generada automáticamente

1. El paciente debe colocar una breve descripción para comprender mejor el caso no es necesario datos personales ya que estarán registrados de forma automática dentro de la aplicación.

Captura de pantalla de un celular

Descripción generada automáticamente

1. Luego sigue seleccionar la fecha y hora que se desee realizar la cita

Captura de pantalla de un celular

Descripción generada automáticamente

1. Al seleccionar la fecha y hora se realiza de la siguiente forma primero seleccionando una fecha en el calendario, luego seleccionando la hora especifica para el inicio de la sesión.

Interfaz de usuario gráfica, Aplicación, Calendario

Descripción generada automáticamente

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

1. Ahora podremos seleccionar de las horas que se necesitan para la cita, las siguientes opciones son las que están disponibles

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

6.Una vez rellenado los campos se pueden enviar la consulta y regresar a la pantalla principal de los pacientes y se reflejaran las nuevas consultas realizadas por el usuario

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## Módulo de aceptar consultas (Doctor)

1. A través del menú de navagación dentro de la aplicación el usuario de doctor puede acceder a esta pantalla donde tendra las consultas creadas por pacientes y sus casos aun no ha sido aceptados

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Para aceptar el caso por parte del usuario de doctor debera presionado la carta del caso por un par de segundos hasta que salga una ventana que podrá confirmar si desea aceptar el caso.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

1. Despues de aceptar el caso el sistema registrara el cambio en la consulta y aparecera en la pantalla de “Proximas Citas”.

## Módulo de próximas Citas (Doctor)

1. En la pantalla de proximas citas para el doctor cuando se inicia la aplicación apareceran las proximas citas que tiene el usuario doctor

Interfaz de usuario gráfica, Texto, Aplicación, Carta

Descripción generada automáticamente

1. Si no estan actualizadas las cita el usuario puede refrescar los datos en pantalla con un simple deslice de dedo hacia abajo en la pantalla actual y se refrescaran las proximas citas.

# Diccionario de la base de datos

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User | | | | | | |
| **ATRIBUTO** | **TIPO DE DATO** | **PK** | **FK** | **OBLI** | **NULL** | **AUTOINC.** |
| Id | String | X |  |  | NO |  |
| username | String |  |  |  | NO |  |
| email | String |  |  |  | NO |  |
| password | String |  |  |  | NO |  |
| Rol | String |  |  |  | NO |  |
| Nombre | String |  |  |  | NO |  |
| estado | Boolean |  |  |  | NO |  |
| rooms | Relacional |  | X |  | NO |  |
| consultas\_doctor | Relacional |  | X |  | NO |  |
| consultas\_paciente | Relacional |  | X |  | NO |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Avatar | | | | | | |
| **ATRIBUTO** | **TIPO DE DATO** | **PK** | **FK** | **OBLI** | **NULL** | **AUTOINC.** |
| Id | String | X |  |  | NO |  |
| Profilepic | Media |  |  |  | NO |  |
| usuario | Relacional |  | X |  | NO |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mensajes | | | | | | |
| **ATRIBUTO** | **TIPO DE DATO** | **PK** | **FK** | **OBLI** | **NULL** | **AUTOINC.** |
| Id | String | X |  |  | NO |  |
| texto | String |  |  |  | NO |  |
| room | Relacional |  | X |  | NO |  |
| From | Relacional |  | X |  | NO |  |
| To | Relacional |  | X |  | NO |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rooms | | | | | | |
| **ATRIBUTO** | **TIPO DE DATO** | **PK** | **FK** | **OBLI** | **NULL** | **AUTOINC.** |
| Id | String | X |  |  | NO |  |
| Chats | Relacional |  | X |  | NO |  |
| Miembros | Relacional |  | X |  | NO |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Consultas | | | | | | |
| **ATRIBUTO** | **TIPO DE DATO** | **PK** | **FK** | **OBLI** | **NULL** | **AUTOINC.** |
| Id | String | X |  |  | NO |  |
| Fecha\_cita | Datetime |  |  |  | NO |  |
| Descripcion | String |  |  |  | NO |  |
| estado | Boolean |  |  |  | NO |  |
| horas | String |  |  |  | NO |  |
| Paciente | Relacional |  | X |  | NO |  |
| Doctor | Relacional |  | X |  | NO |  |