**DESARROLLO DE UN BOT TELEGRAM PARA AUTOMATIZAR EL REGISTRO DE ASISTENCIA DE ESTUDIANTES UNIVERSITARIOS A CLASES**



**JOSE UBALDO CARVAJAL**

**Estudiante maestría gestión y desarrollo de proyectos de software**

**Profesores**

**Sandra Victoria Hurtado Gil**

**Jorge Iván Meza Martinez**

**UNIVERSIDAD AUTÓNOMA DE MANIZALES**

**MAESTRIA EN GESTIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE**

**LINEA DE INVESTIGACIÓN EN CALIDAD DE SOFTWARE**

**ÉNFASIS EN DESARROLLO DE SOFTWARE**

**MANIZALES**

**2018**

Contenido

[Contenido 2](#_Toc533366919)

[1. INTRODUCCIÓN, PROPÓSITO DEL TRABAJO 6](#_Toc533366920)

[2. TECNOLOGÍAS/HERRAMIENTAS TÉCNICAS UTILIZADAS 7](#_Toc533366921)

[3. INTRODUCCIÓN A TELEGRAM 8](#_Toc533366922)

[4. DESCRIPCIÓN DEL TRABAJO A REALIZAR 9](#_Toc533366923)

[4.1. Objetivos del trabajo 9](#_Toc533366924)

[4.2. Historia de Usuario “Visualizar menú principal Estudiante” 10](#_Toc533366925)

[4.2.1. Descripción 10](#_Toc533366926)

[4.2.2. Prototipo no funcional 10](#_Toc533366927)

[4.3. Historia de Usuario “Actualizar datos básicos estudiante” 11](#_Toc533366928)

[4.3.1. Descripción 11](#_Toc533366929)

[4.3.2. Prototipo no funcional 11](#_Toc533366930)

[4.4. Historia de Usuario “Inscribir asignatura” 13](#_Toc533366931)

[4.4.1. Descripción 13](#_Toc533366932)

[4.4.2. Prototipo no funcional 14](#_Toc533366933)

[4.5. Historia de Usuario “Registrar asistencia” 15](#_Toc533366934)

[4.5.1. Descripción 15](#_Toc533366935)

[4.5.2. Prototipo no funcional 15](#_Toc533366936)

[4.6. Historia de Usuario “Reporte asistencia asignatura” 16](#_Toc533366937)

[4.6.1. Descripción 16](#_Toc533366938)

[4.6.2. Prototipo no funcional 17](#_Toc533366939)

[4.7. Historia de Usuario “Geo-referenciar aula” 18](#_Toc533366940)

[4.7.1. Descripción 18](#_Toc533366941)

[4.7.2. Prototipo no funcional 19](#_Toc533366942)

[5. ARQUITECTURA DE LA APLICACIÓN BOT 20](#_Toc533366943)

[6. MODELO DE DOMÍNIO DE APLICACIÓN BOT 21](#_Toc533366944)

[7. PROCESO DE CONSTRUCCIÓN DEL BOT 22](#_Toc533366945)

[7.1. Creación/registro del *Bot* en Telegram 22](#_Toc533366946)

[7.2. Creación de aplicación *Heroku* en *NodeJS* 23](#_Toc533366947)

[7.3. Codificación del *Bot* 24](#_Toc533366948)

[7.3.1. *TypeScript* 24](#_Toc533366949)

[7.3.2. *API Telegram* 25](#_Toc533366950)

[7.3.3. Librería *node-telegram-bot-api* 26](#_Toc533366951)

[7.3.4. Base de datos *Firebase* 27](#_Toc533366952)

[7.3.5. Estructura de la aplicación *Bot* 28](#_Toc533366953)

[7.3.6. Publicación de una versión funcional del Bot 29](#_Toc533366954)

[7.3.7. Pantallazos aplicación *Bot* 30](#_Toc533366955)

[8. LISTA DE REFERENCIA 36](#_Toc533366956)

[9. ANEXOS 37](#_Toc533366957)

**Lista de Figuras**

Figura 1. Bot DeLorean para gestionar recordatorios. Fuente: https://recast.ai/blog/top-telegram-bots-2017/ 8

Figura 2. Prototipo historia de usuario “Visualizar menú principal Estudiante” 10

Figura 3. Prototipo historia de usuario “Actualizar datos básicos estudiante” 12

Figura 4. Prototipo historia de usuario “Inscribir asignatura” 14

Figura 5. Prototipo historia de usuario “Registrar asistencia” 16

Figura 6. Prototipo historia de usuario “Consultar inasistencias para asignatura” 17

Figura 7. Prototipo historia de usuario “Geo-referenciar aula” 19

Figura 8. Arquitectura del *Bot* para registrar asistencia a clases 20

Figura 9. Modelo de domínio del *Bot* para registrar asistencia a clases 21

Figura 10. *Botfather*, utilidad para crear y configurar *Bots* en *Telegram.* 22

Figura 11. *Botfather*, utilidad para crear y configurar *Bots* en *Telegram.* 23

Figura 12. *Playground TypeScript.* 24

Figura 13. *Telegram API.* 25

Figura 14. Librería *node-telegram-bot-api*. 26

Figura 15. *Firebase*. 27

Figura 16.Estructura del proyecto, parte I. 28

Figura 17.Estructura del proyecto, parte II. 28

Figura 18.Despliegue del proyecto, parte I. 29

Figura 19.Despliegue del proyecto, parte II. 29

Figura 20.Pantalla bienvenida *Bot*. 30

Figura 21.Menú estudiante. 30

Figura 22.Inscribir asignatura, parte I. 31

Figura 23.Inscribir asignatura, parte II. 31

Figura 24.Inscribir asignatura, parte III. Aprobación docente. 32

Figura 25.Registrar asistencia parte I. 32

Figura 26.Registrar asistencia parte II, solicitud autorización GPS. 33

Figura 27.Registrar asistencia parte III, envío de coordenadas GPS. 33

Figura 28.Generar reporte asistencia I. 34

Figura 29.Generar reporte asistencia II. 34

Figura 30.Generar reporte asistencia III. 35

Figura 31.Generar reporte asistencia IV. 35

**Lista de Tablas**

Tabla 1. Tecnologías/herramientas utilizadas. 7

Tabla 2. Historia de usuario “Visualizar menú principal Estudiante”. 10

Tabla 3. Historia de usuario “Actualizar datos básicos estudiante”. 11

Tabla 4. Historia de usuario “Inscribir asignatura”. 14

Tabla 5. Historia de usuario “Registrar asistencia en asignatura”. 15

Tabla 6. Historia de usuario “Consultar inasistencias para asignatura”. 17

Tabla 7. Historia de usuario “Consultar inasistencias para asignatura”. 18

1. INTRODUCCIÓN, PROPÓSITO DEL TRABAJO

En el presente trabajo se pretende desarrollar un Bot para Telegram con objeto de automatizar el registro de asistencia de estudiantes universitarios a clases.

En este documento pretende servir de guía para ilustrar de forma general las actividades necesarias para desarrollar un *Bot* *Telegram*.

La implementación del proyecto sobre el que se aplicaron los conceptos y herramientas se puede encontrar en el repositorio Github: <https://github.com/joseucarvajal/profeJoseBotFinal>.

1. TECNOLOGÍAS/HERRAMIENTAS TÉCNICAS UTILIZADAS

|  |  |  |
| --- | --- | --- |
| **Tecnología/Herramienta** | **Versión** | **Fuente/Descarga** |
| Telegram Desktop | 1.5.3 | <https://desktop.telegram.org/> |
| Telegram para Android | 5.1 | <https://play.google.com/store/apps/details?id=org.telegram.messenger&hl=en> |
| NodeJS | 10.14.2 | https://nodejs.org/en/download/ |
| Visual Studio Code | 1.30.1 | <https://code.visualstudio.com/download> |
| Git Client | 2.20.1 | <https://git-scm.com/downloads> |
| TypeScript | 2.4.2 | Vía npm (Node Package Manager): **npm install -g typescript** |
| Subscripción gratuita Heroku para desplegar el Bot. |  | https://www.heroku.com |
| API Telegram | 4.1 | https://core.telegram.org/bots/api |

Tabla 1. Tecnologías/herramientas utilizadas.

1. INTRODUCCIÓN A TELEGRAM

[Telegram](https://telegram.org/) es una plataforma para mensajería instantánea que incluye una aplicación para los principales sistemas operativos móviles (Android y iOS). Adicionalmente, incluye un cliente escritorio y uno web. Las principales características de Telegram es que las conversaciones son privadas y cifradas, es basado en la nube, es gratis… entre otras (Telegram, 2018). Telegram es uno de los principales competidores de la aplicación [WhatsApp](https://www.whatsapp.com/). Una de las características fundamentales de Telegram, es su capacidad de extensibilidad, en tato permite desarrollar *Bots* que son programas autónomos que pueden interactuar con sistemas o usuarios, de esta manera un usuario de Telegram puede instalar e interactuar con un *Bot* que le presente las noticias. Un ejemplo de aplicación *Bot* es [DeLorean](https://telegram.me/delorean_bot), este *Bot* permite gestionar recordatorios y notificaciones para eventos importantes.

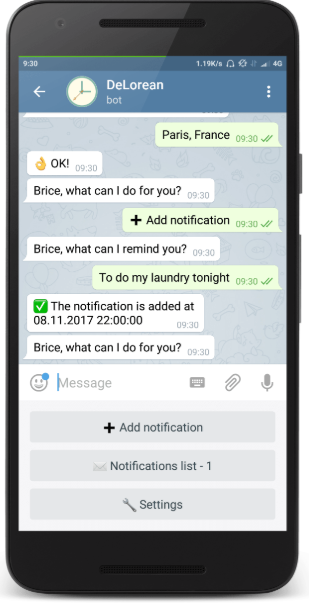


Figura 1. Bot DeLorean para gestionar recordatorios.   
Fuente: <https://recast.ai/blog/top-telegram-bots-2017/>

Los *Bots* se convierten como un contacto más en Telegram de tal manera que los usuarios pueden interactuar con ellos.

1. DESCRIPCIÓN DEL TRABAJO A REALIZAR

En este trabajo se ilustra de manera general el proceso técnico para la creación y desarrollo de un *Bot* para *Telegram*. Para esto, se construirá un Bot denominado **ProfeJoseBot** que servirá para automatizar el registro de asistencia de estudiantes a clases. Este trabajo se realiza con fines académicos y pretende servir como guía técnica básica a la hora de crear un *Bot* para *Telegram*, por lo cual sus funcionalidades y alcances han sido delimitados y no constituye un proyecto o producto para el sector educativo.

En este apartado se esbozará el alcance del trabajo, las características, casos de uso principales y arquitectura de la aplicación *Bot* para *Telegram*.

* 1. Objetivos del trabajo
* Documentar el proceso general para el desarrollo de un *Bot* para *Telegram*.
* Desarrollar un *Bot* que permita registrar la asistencia de estudiantes a clases.

A continuación, se presentan las historias de usuario que se cubrirán en el desarrollo del *Bot* para registro de asistencia a clases:

* 1. Historia de Usuario “Visualizar menú principal Estudiante”
     1. Descripción

Una vez el estudiante ha agregado el **profeJoseBot** en sus contactos, tiene acceso al menú principal.

|  |  |
| --- | --- |
| **Visualizar menú principal Estudiante** | |
| **Descripción** | **Criterios de aceptación** |
| El estudiante visualiza el menú principal con las siguientes opciones:   1. Registrar asistencia. 2. Inscribir asignatura. 3. Actualizar mis datos básicos. | 1. El estudiante puede seleccionar una de las opciones del menú. |

Tabla 2. Historia de usuario “Visualizar menú principal Estudiante”.

* + 1. Prototipo no funcional

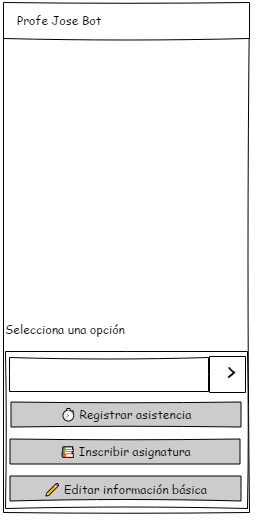


Figura 2. Prototipo historia de usuario “Visualizar menú principal Estudiante”

* 1. Historia de Usuario “Actualizar datos básicos estudiante”
     1. Descripción

Permite a un estudiante editar sus datos básicos en el *Bot*. Un estudiante debe actualizar sus datos básicos antes de proceder a realizar alguna otra acción como “Registrar asistencia” o “Registrar asignatura”. Este paso puede omitirse cuando el estudiante ya se encuentra en el listado de asignaturas del docente provisto por Registro Académico.

|  |  |
| --- | --- |
| **Actualizar datos básicos estudiante** | |
| **Descripción** | **Criterios de aceptación** |
| Luego de instalar el Bot en su cuenta de *Telegram*, un estudiante procede a actualizar sus datos básicos:   1. Código estudiantil: código del estudiante asignado por la universidad (Requerido). 2. Nombre completo: Nombre completo del estudiante (Requerido). 3. Email: Correo electrónico del estudiante. (Requerido). | 1. El estudiante ingresa los valores requeridos. 2. Una vez el estudiante se ha registrado exitosamente el Bot le indica que ha realizado el proceso satisfactoriamente y se despliega el menú de opciones nuevamente. |

Tabla 3. Historia de usuario “Actualizar datos básicos estudiante”.

* + 1. Prototipo no funcional

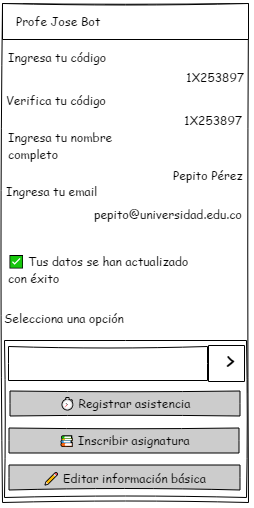


Figura 3. Prototipo historia de usuario “Actualizar datos básicos estudiante”

* 1. Historia de Usuario “Inscribir asignatura”
     1. Descripción

Una vez el estudiante ha ingresado sus datos básicos o bien se encuentre en el listado de asistencia de asignaturas provisto por Registro Académico, debe confirmar las materias que está viendo con el docente, si el estudiante no se encuentra en el listado de alguna de las asignaturas asociadas al profesor, puede enviar una solicitud al docente, una vez el docente apruebe esta solicitud, el estudiante podría registrar asistencia en dicha materia. Esto se realiza para escenarios donde los estudiantes aún no están matriculados en alguna clase específica.

|  |  |
| --- | --- |
| **Inscribir asignatura** | |
| **Descripción** | **Criterios de aceptación** |
| Un estudiante ingresa al *Bot* y selecciona la opción “📚 **Inscribir asignatura**”. El Bot le muestra un listado de asignaturas en las que el estudiante se encuentra matriculado y le ofrece dos opciones, confirmar las asignaturas que ya se encuentra matriculado o solicitar la inscripción en alguna de las asignaturas que no se encuentre matriculado:   1. El estudiante selecciona la opción “📚 **Inscribir asignatura**”. 2. El sistema despliega un listado de asignatura correspondientes al periodo actual en las que el estudiante se encuentra matriculado. 3. El estudiante confirma que el listado de asignaturas que despliega el Bot corresponde a las materias que se encuentra cursando con el profesor. 4. El Bot internamente asociará al estudiante con las asignaturas seleccionadas. 5. En el caso que el estudiante no se encuentre matriculado a las asignaturas que orienta el docente. El Bot gestionará la comunicación para que el estudiante pueda enviarle una solicitud al docente de inscripción de asignatura. El Bot le envía esta solicitud al docente quien puede aceptarla o rechazarla. Cualquiera sea el caso, el Bot notificará al estudiante una vez el docente haya decidido aprobar o no la solicitud de ingresar asistencia para una asignatura en particular. | 1. El sistema asocia al estudiante en las asignaturas que el alumno confirmó estar matriculado. Según información de Registro Académico. 2. Si el estudiante no aparece matriculado en ninguna asignatura. Podrá elegir una materia de las que orienta el docente y enviar una solicitud de inscripción. 3. Si el docente aprueba la solicitud de inscripción del estudiante, el alumno podrá registrar en la asignatura. De lo contrario no podrá hacerlo. |

Tabla 4. Historia de usuario “Inscribir asignatura”.

* + 1. Prototipo no funcional

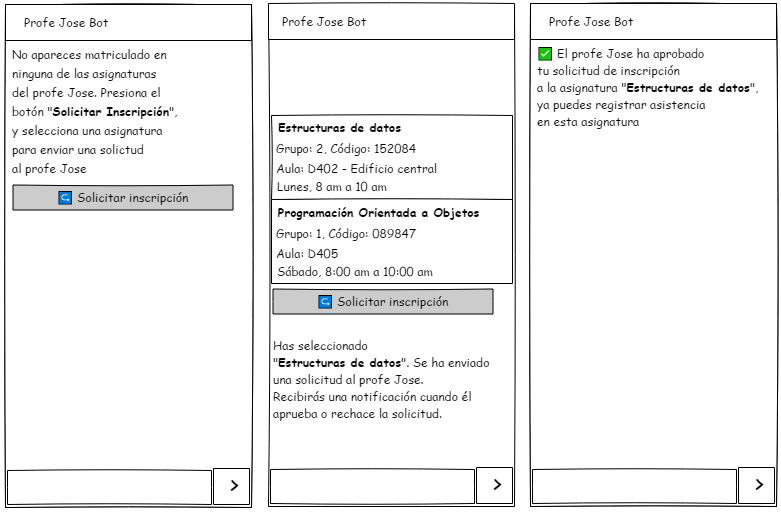


Figura 4. Prototipo historia de usuario “Inscribir asignatura”

* 1. Historia de Usuario “Registrar asistencia”
     1. Descripción

Cuando un estudiante desea registrar asistencia a una de las asignaturas a las que se encuentra inscrito o matriculado.

|  |  |
| --- | --- |
| **Registrar asistencia en asignatura** | |
| **Descripción** | **Criterios de aceptación** |
| Un estudiante ingresa al *Bot* y selecciona la opción “**⏱ Registrar asistencia**”. A continuación, el estudiante selecciona una asignatura de un listado y registra asistencia.   1. El estudiante selecciona la opción “**⏱ Registrar asistencia**”. 2. El sistema despliega un listado de asignaturas que tiene registradas el estudiante. 3. El estudiante selecciona una asignatura y el sistema registra la asistencia. | 1. El sistema captura la ***asignatura, fecha, hora y coordenadas espaciales*** donde se encuentra el estudiante y registra estos datos en la asistencia. 2. Si las coordenadas GPS del estudiante no se encuentran dentro de un radio establecido con respecto al centro del aula de clase, el *Bot* no permite registrar asistencia y le informa al estudiante lo sucedido. 3. Si la hora en la que el estudiante desea registrar asistencia no se encuentra dentro del rango del horario de la asignatura, el *Bot* no permite registrar asistencia y le informa al estudiante lo sucedido. |

Tabla 5. Historia de usuario “Registrar asistencia en asignatura”.

* + 1. Prototipo no funcional

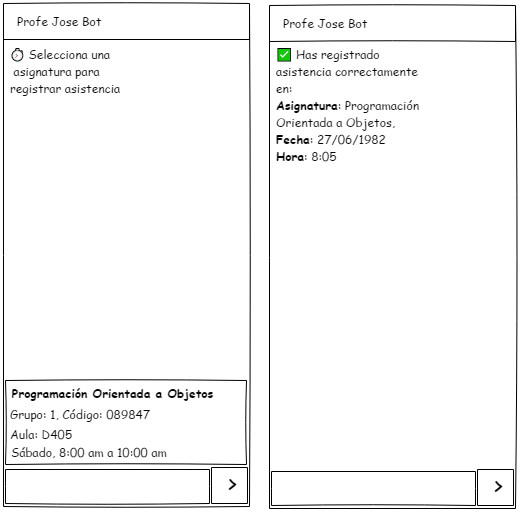


Figura 5. Prototipo historia de usuario “Registrar asistencia”

* 1. Historia de Usuario “Reporte asistencia asignatura”
     1. Descripción

Le da la posibilidad al profesor de consultar las asistencias e inasistencias de una asignatura en particular.

|  |  |
| --- | --- |
| **Consultar inasistencias para asignatura** | |
| **Descripción** | **Criterios de aceptación** |
| El profesor ingresa al *Bot* y selecciona la opción “**📆 Reporte asistencia asignatura**”. A continuación, el docente selecciona una asignatura de un listado y obtiene un reporte de asistencias e inasistencias para la misma.   1. El docente selecciona la opción “**📆 Reporte asistencia asignatura**”. 2. El sistema despliega un listado de asignaturas que tiene asociadas el profesor. 3. El docente selecciona una asignatura. 4. El sistema despliega los datos de inasistencias registrados para la asignatura. | 1. El sistema valida la identidad del docente según su número celular. 2. Si el número celular del docente no corresponde, el sistema informa de lo sucedido y bloquea el acceso a las opciones del profesor. 3. El Bot le envía al docente un reporte en formato PDF con los siguientes datos:    * Datos básicos de la asignatura: nombre, código, grupo y horarios.    * Datos básicos del estudiante: código, nombre, email.    * Número de asistencias a la fecha.    * Número de fallas a la fecha. |

Tabla 6. Historia de usuario “Consultar inasistencias para asignatura”.

* + 1. Prototipo no funcional

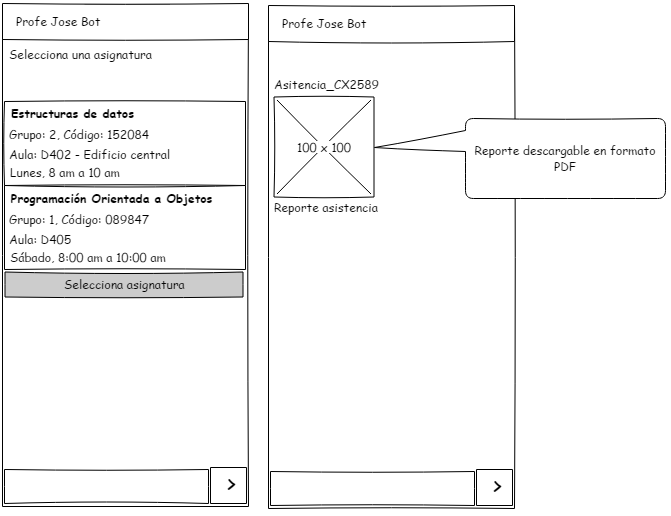


Figura 6. Prototipo historia de usuario “Consultar inasistencias para asignatura”

* 1. Historia de Usuario “Geo-referenciar aula”
     1. Descripción

Le permite al docente capturar las coordenadas GPS de un aula específica para que la toma de asistencia de estudiantes se haga con respecto a estas coordenadas y un radio dado. Esto permite que si se cambia el salón de clases de una asignatura, el docente pueda actualizar las coordenadas del nuevo salón en tiempo real.

|  |  |
| --- | --- |
| **Georeferenciar aula** | |
| **Descripción** | **Criterios de aceptación** |
| El profesor ingresa al *Bot* y selecciona la opción “**📆 Reporte asistencia asignatura**”. A continuación, el docente selecciona una asignatura de un listado y obtiene un reporte de asistencias e inasistencias para la misma.   1. El docente selecciona la opción “**🌎 Georeferenciar aula**”. 2. El sistema despliega un listado de asignaturas que tiene asociadas el profesor. 3. El docente selecciona la asignatura que desea geo-referenciar. 4. El sistema captura las coordenadas GPS y las asocia con la asignatura. | 1. El sistema valida la identidad del docente según su número celular. 2. Si el número celular del docente no corresponde, el sistema informa de lo sucedido y bloquea el acceso a las opciones del profesor. 3. El Bot solicita permiso para acceder a las coordenadas GPS del profesor y asocia estas coordenadas a la asignatura seleccionada por el docente. |

Tabla 7. Historia de usuario “Consultar inasistencias para asignatura”.

* + 1. Prototipo no funcional

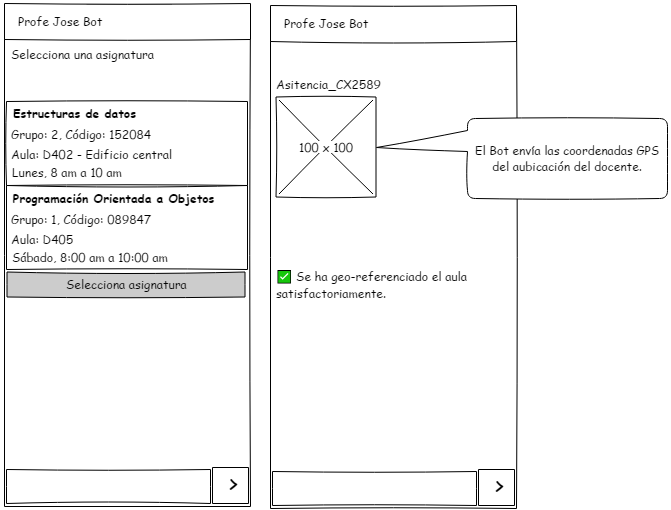


Figura 7. Prototipo historia de usuario “Geo-referenciar aula”

1. ARQUITECTURA DE LA APLICACIÓN BOT

En este apartado del documento se ilustra la arquitectura de la aplicación *Bot*:

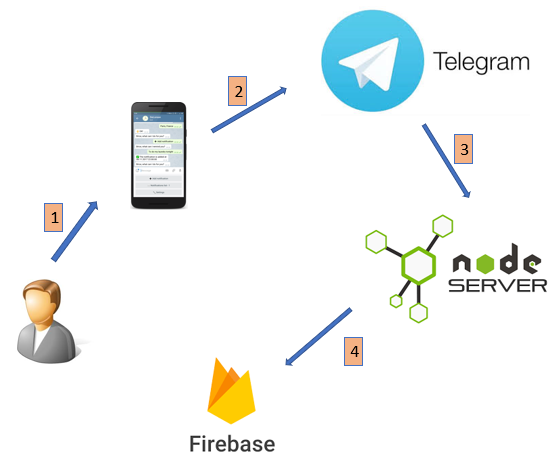


Figura 8. Arquitectura del *Bot* para registrar asistencia a clases

Inicialmente, en el punto 1, un usuario busca e incluye el *Bot* en sus contactos, de esta manera, podrá interactuar con las funcionalidades ofrecidas por el *Bot*,

En el segundo punto, el mensaje es enviado a los servidores de *Telegram*.

En el tercer punto, *Telegram* establece conexión con el servidor *NodeJS*, donde reside la lógica *Back-End* del *Bot*. *Telegram* le envía el mensaje que proviene desde el usuario al servidor *NodeJS*. Adicionalmente, *Telegram* agrega una información adicional al mensaje (chat-id, autor del mensaje… entre otros).

Finalmente, en el paso 4, el servidor *back-end* *NodeJS* establece contacto con el repositorio de datos que en este caso será *Firebase.*

1. MODELO DE DOMÍNIO DE APLICACIÓN BOT

Para la persistencia de datos del *Bot*, se utilizará una base de datos NoSQL de Google denomina *Firebase*. A continuación, se muestra el modelo de dominio de la aplicación:

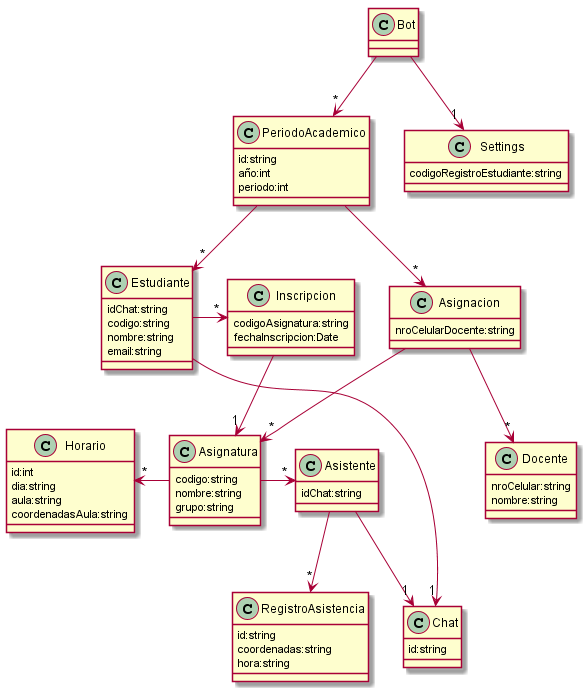


Figura 9. Modelo de domínio del *Bot* para registrar asistencia a clases

1. PROCESO DE CONSTRUCCIÓN DEL BOT

A continuación, se explica el proceso de construcción del *Bot* y se listan las referencias de cada uno de los pasos del proceso.

* 1. Creación/registro del *Bot* en Telegram

Para empezar, se debe registrar/crear el *Bot* en *Telegram*. Para esto se utiliza *Botfather*, un *Bot* creado por *Telegram* para crear/registrar *Bots.* Los pasos para crear un *Bot* a través de *Botfather* se pueden consultar en: <https://core.telegram.org/bots>.

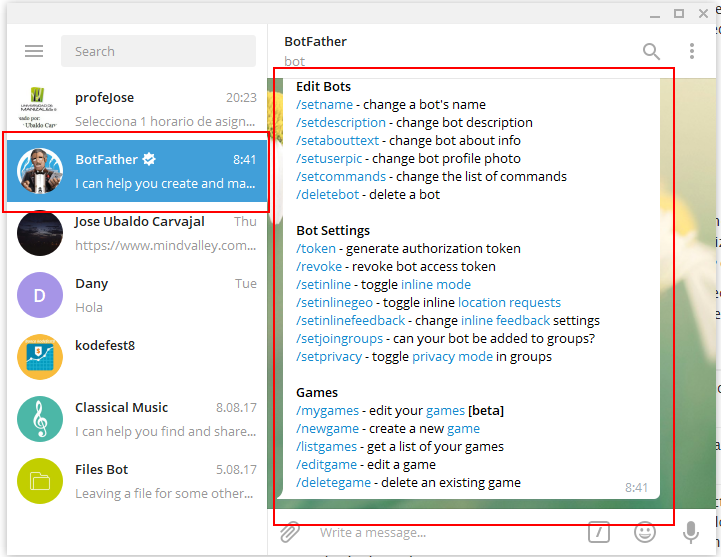


Figura 10. *Botfather*, utilidad para crear y configurar *Bots* en *Telegram.*

* 1. Creación de aplicación *Heroku* en *NodeJS*

Una vez se ha creado el *Bot* a través de *Botfather*, se procede a crear la infraestructura (servidor) para desplegar el *Bot*. Para esto, se utilizó la plataforma *Heroku* (<https://www.heroku.com>) que es un servicio en la nube para crear y hospedar aplicaciones en diferentes plataformas (*Java, PHP, NodeJS*… etc.). *Heroku* ofrece una subscripción gratuita con limitantes en recursos, pero suficiente para desplegar la solución del *Bot*. En el enlace <https://devcenter.heroku.com/articles/getting-started-with-nodejs>. se encuentran los pasos detallados para crear una aplicación gratis en *Heroku*, basada en *NodeJS*.

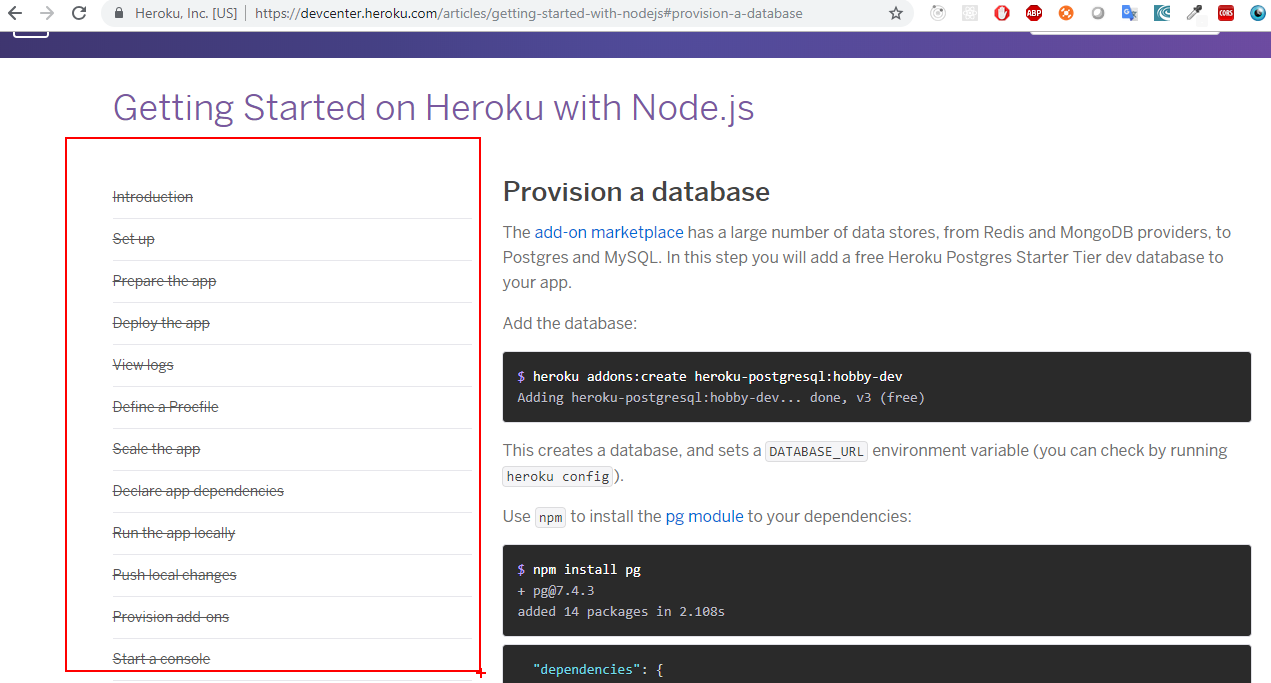


Figura 11. *Botfather*, utilidad para crear y configurar *Bots* en *Telegram.*

* 1. Codificación del *Bot*
     1. *TypeScript*

En primera instancia, se utilizó *TypeScript* como lenguaje de programación, dado que permite generar/transpilar código JavaScript. La ventaja de *TypeScript* es que es un lenguaje fuertemente tipado, lo que facilita la re-factorización del código y evita cometer errores colaterales por falta de tipado en *JavaScript*. *TypeScript* se puede descargar desde el siguiente enlace: [https://www.typescriptlang.org/index.html#download-links](https://www.typescriptlang.org/index.html%23download-links).Adicionalmente, *TypeScript* ofrece una página donde se puede probar código escrito en *TypeScript*. <https://www.typescriptlang.org/play/index.html>.

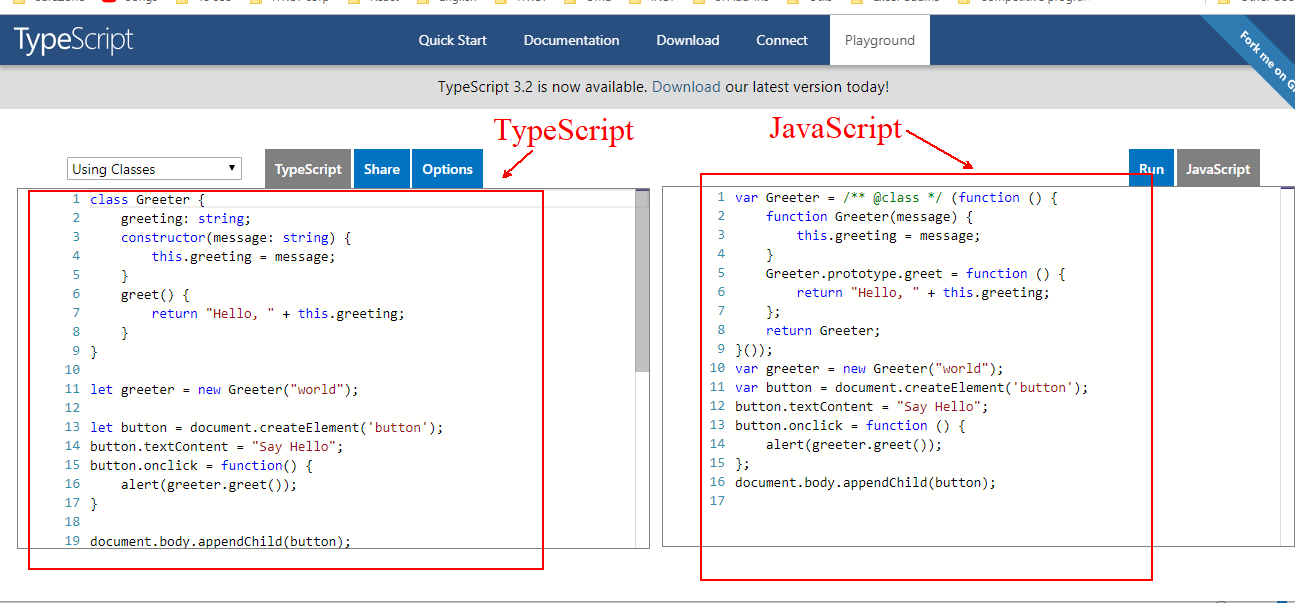


Figura 12. *Playground TypeScript.*

* + 1. *API Telegram*

Seguidamente, se debe tener claro el funcionamiento del API de *Telegram*. Esto se puede consultar en el siguiente link: <https://core.telegram.org/bots/api>.

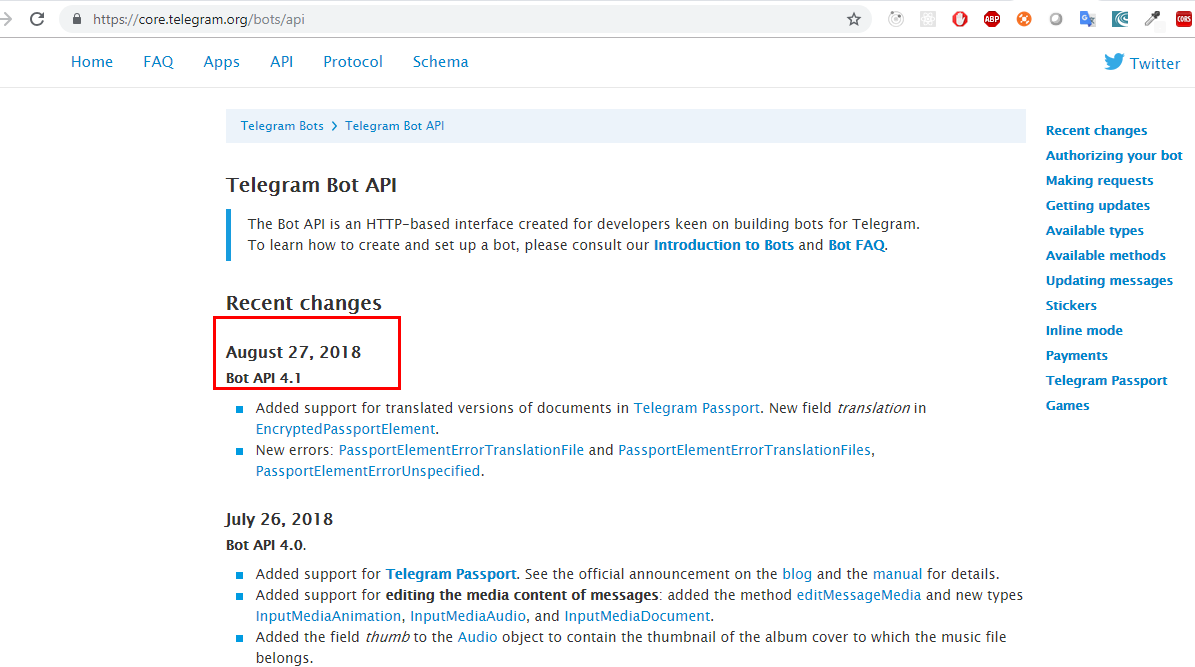


Figura 13. *Telegram API.*

* + 1. Librería *node-telegram-bot-api*

Posteriormente, se debe instalar la librería **node-telegram-bot-api**, que provee una abstracción sobre el API de *Telegram* en NodeJS para facilitar la codificación del *Bot*. En el siguiente enlace se puede descargar la librería **node-telegram-bot-api**: <https://github.com/yagop/node-telegram-bot-api>.

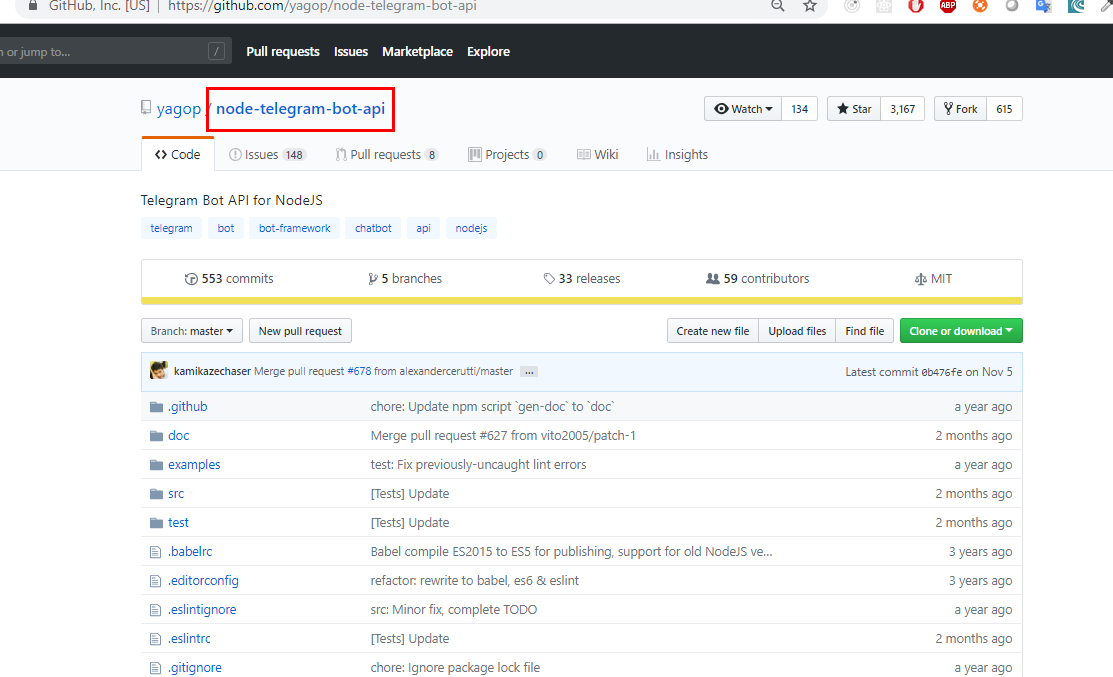


Figura 14. Librería *node-telegram-bot-api*.

* + 1. Base de datos *Firebase*

Para la gestión de datos que produce y consume el *Bot* se puede utilizar *Firebase* que es un servicio de base de datos *NoSQL* provisto por la compañía *Google*. *Firebase* tiene un plan gratuito para creación y manipulación de datos, este plan, al ser gratuito, tiene algunas limitantes como número de transacciones por hora… etc. Se puede consultar más información sobre *Firebase* en la siguiente página: <https://firebase.google.com/>. En el siguiente enlace se puede encontrar un tutorial de *Firebase:* <https://www.tutorialspoint.com/firebase/>.

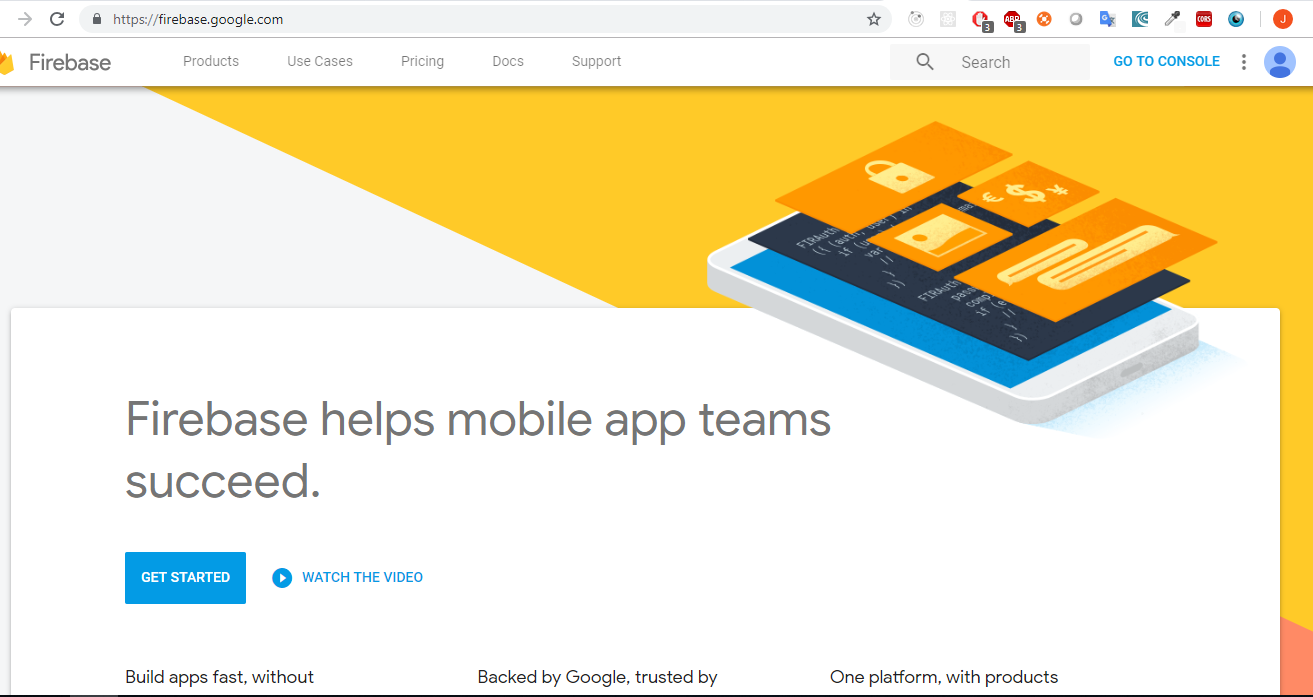


Figura 15. *Firebase*.

* + 1. Estructura de la aplicación *Bot*

En este apartado se ilustra la estructura de carpetas y archivos que conforman la solución.

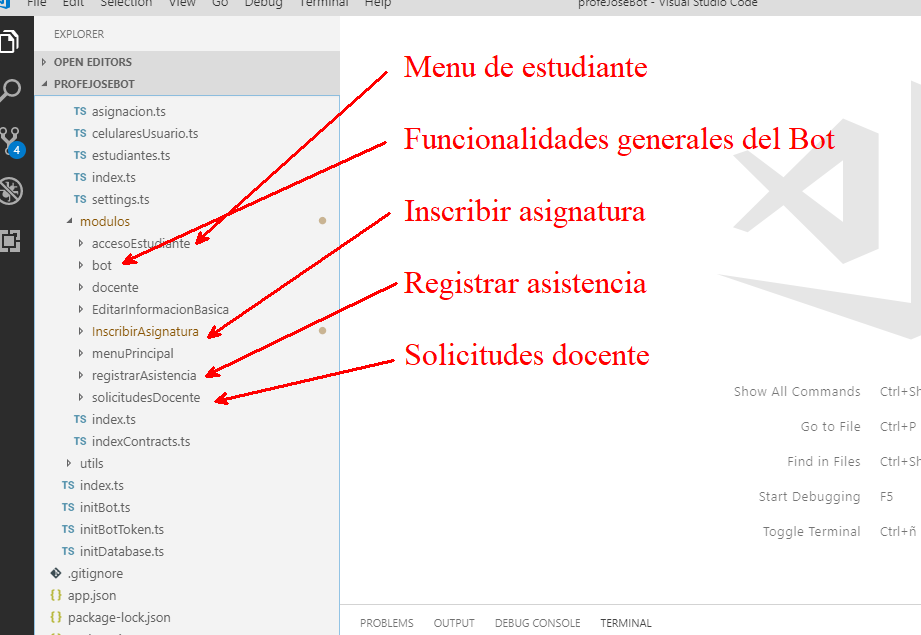


Figura 16.Estructura del proyecto, parte I.

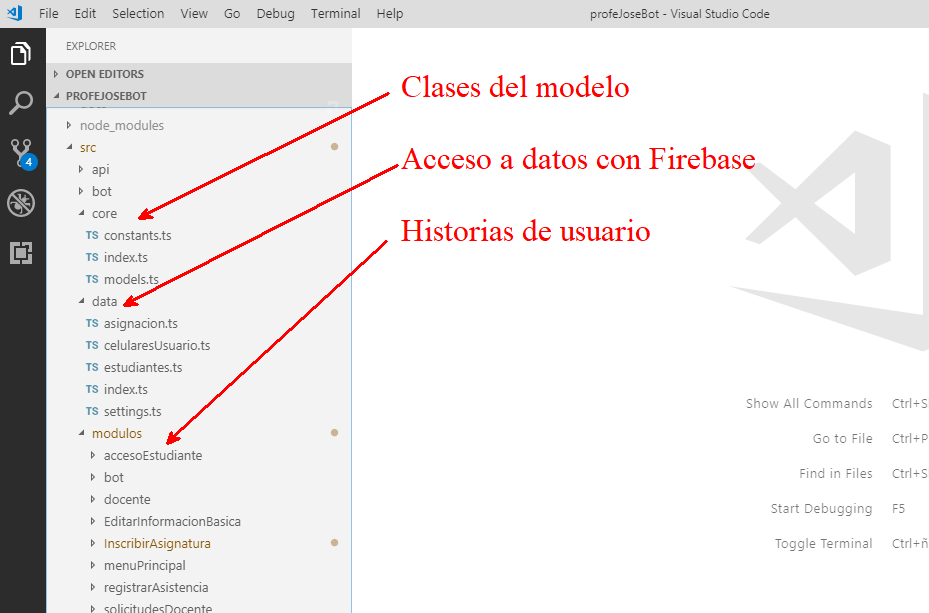


Figura 17.Estructura del proyecto, parte II.

* + 1. Publicación de una versión funcional del Bot

Para publicar una versión funcional del *Bot* en internet, *Heroku* ofrece un servicio de integración con *GitHub*, de tal forma que se puede tomar una versión de la aplicación en *GitHub*, y publicarla en *Heroku* con un solo clic.

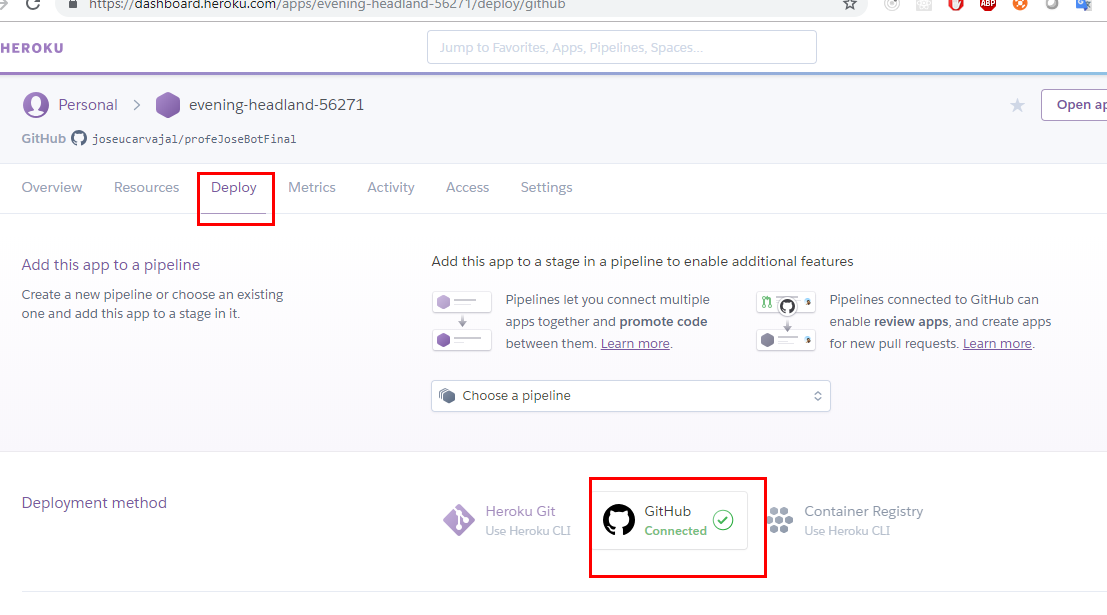


Figura 18.Despliegue del proyecto, parte I.

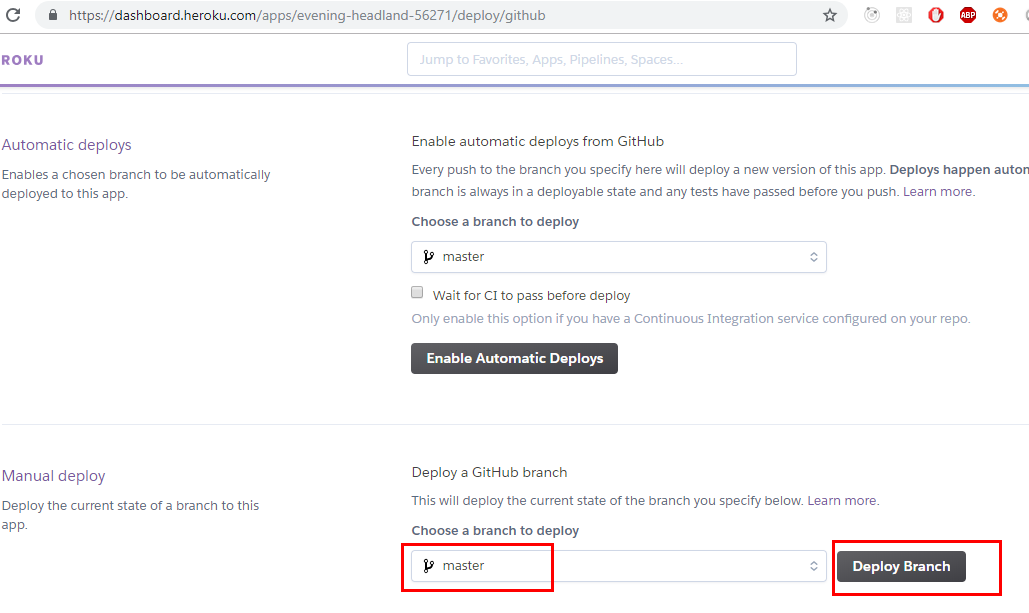


Figura 19.Despliegue del proyecto, parte II.

* + 1. Pantallazos aplicación *Bot*

Finalmente, el *Bot* tiene el siguiente aspecto (versión escritorio de *Telegram*).

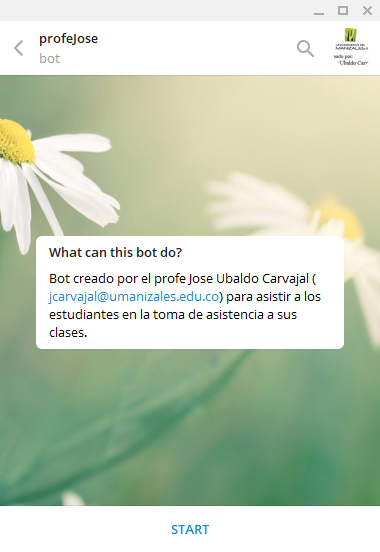


Figura 20.Pantalla bienvenida *Bot*.

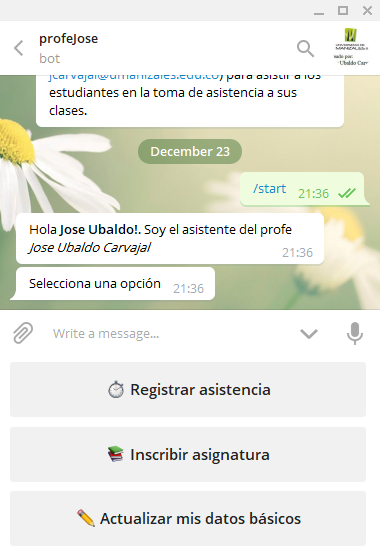


Figura 21.Menú estudiante.

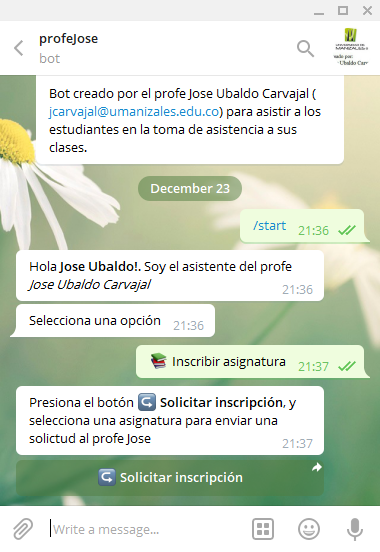


Figura 22.Inscribir asignatura, parte I.

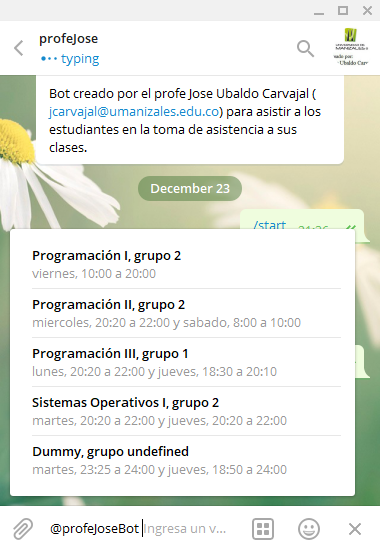


Figura 23.Inscribir asignatura, parte II.

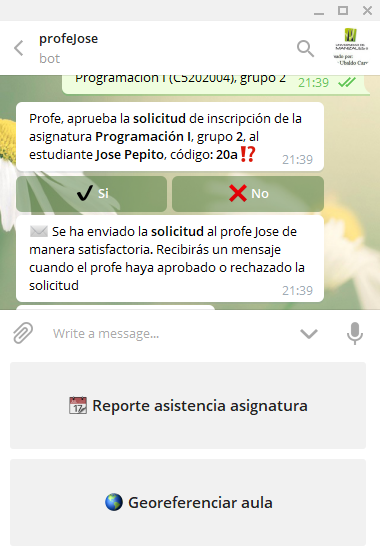


Figura 24.Inscribir asignatura, parte III. Aprobación docente.



Figura 25.Registrar asistencia parte I.

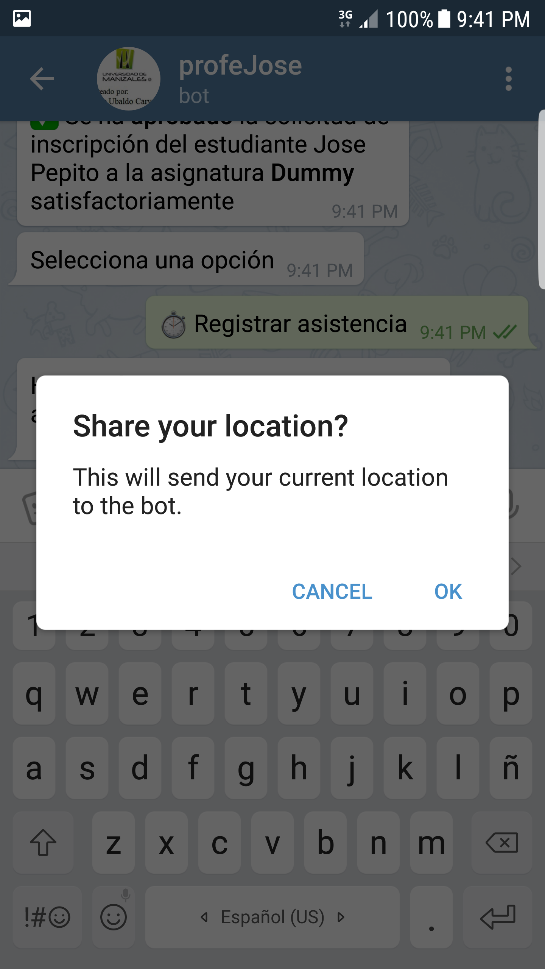


Figura 26.Registrar asistencia parte II, solicitud autorización GPS.

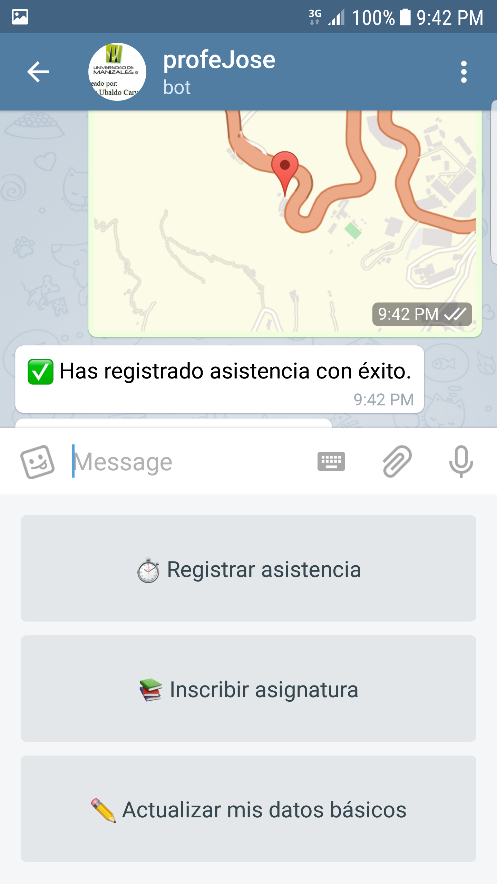


Figura 27.Registrar asistencia parte III, envío de coordenadas GPS.

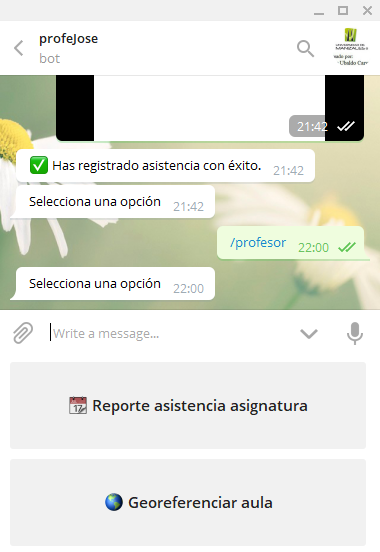


Figura 28.Generar reporte asistencia I.

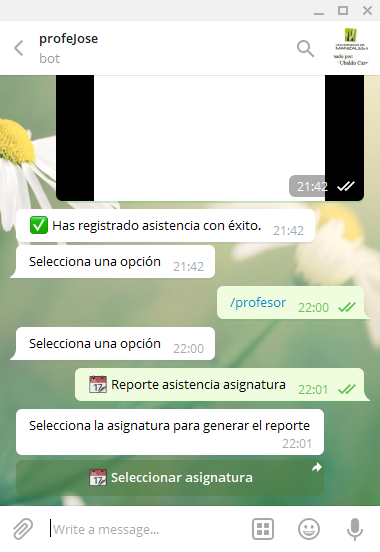


Figura 29.Generar reporte asistencia II.



Figura 30.Generar reporte asistencia III.



Figura 31.Generar reporte asistencia IV.

1. LISTA DE REFERENCIA

Telegram Official documentation. (2018). Why switch to *Telegram*?. Recuperado de <https://telegram.org/>.

Getting Started on Heroku with Node.js. (2018). Heroku. Recuperado de <https://devcenter.heroku.com/articles/getting-started-with-nodejs>.

1. ANEXOS

ANEXO 2. Instancia JSON de ejemplo de la base de datos.