Ya que, en el capítulo x se determinó que la base datos que cumplía todo los requisitos técnicos y funcionales para este proyecto es Maria DB y de la misma forma en el capítulo x se precisó que el servicio de bases de datos en la nube más apto y con mejores beneficios para este proyecto es Amazon RDS (Amazon Relational Database Services), se procedió a implementar de la base de datos en este, por lo cual lo primero que se tomo en cuenta fue que no era conveniente limitar las direcciones IP desde las cuales se podría acceder a este servicio, ya que es necesario poder consultar e insertar datos en la base de datos desde el Gateway de la red de sensores y así mismo poder consultar y modificar los registros que allí se encuentran almacenados desde la aplicación web. Por motivos de seguridad no se realizó la conexión a la base de datos de forma directa desde la aplicación web, ya que las credenciales para acceder a esta quedarían a disposición de cualquier persona que tuviera un mínimo conocimiento de desarrollo web. Por tal razón se opto por utilizar el servicio Amazon API Gateway, el cual permite crear, publicar, mantener, monitorizar y proteger las API fácilmente, dando como resultado una puerta delantera para que las aplicaciones puedan acceder a la información o lógica de negocio que se encuentre alojada en servicios de backend.[[1]](#footnote-1)

El servicio Amazon API Gateway trabaja de la mano con el servicio AWS Lamba, el cual es un servicio que nos permite ejecutar código sin aprovisionar o administrar servidores, por el cual sólo se paga el tiempo de computación que este consuma.[[2]](#footnote-2) Estos dos servicios fueron fundamentales para dar solución a la problemática de acceso seguro a la base de datos desde la aplicación web, ya que el servicio Amazon API Gateway brinda una URL a la cual la aplicación puede realizar una petición. Cuando la aplicación realiza dicha petición al servicio Amazon API Gateway, se encarga de ejecutar el código que se encuentra almacenado en una instancia de AWS Lamba, en nuestro caso un script realizado en Python, el cual contiene de forma segura y privadas las credenciales de la base de datos y se encarga de realizar las consultas o modificaciones en las sentencias, dependiendo de la URL desde la cual se halla realizado la petición. Estas consultas se realizan por medio de la librería pymysql, la cual nos permite realizar la conexión a una base de datos que soporte el lenguaje SQL y de la misma forma ejecutar diversas sentencias en este lenguaje.

Luego de tener todos los elementos disponibles por parte del servidor para acceder a la información desde la aplicación, se procedió a comenzar el desarrollo de esta con el framework react.js, framework que se seleccionó previamente en el capitulo x con ayuda del cuadro comparativo x. En primer lugar, se procedió a instalar las dependencias necesarias para llevar a cabo el desarrollo, dentro de los cuales se encuentra node.js el cual un entorno de ejecución de JavaScript y está diseñado para crear aplicaciones network escalables.[[3]](#footnote-3) Esta instalación se llevo a cabo por medio del comando: npm install -g npm@latest, luego de poseer esta dependía se procedió a crear el proyecto de react.js por medio del siguiente comando: create-react-app Xiscua. Este comando nos brinda una estructura inicial de un proyecto, la cual se modifica según las necesidades para llevar a cabo el desarrollo de cada una de las clases que se ilustraron en el diagrama de clases. Con ayuda de Bootstrap, el cual es el kit de herramientas open source más popular de todo el mundo [[4]](#footnote-4), se agilizó el desarrollo de la aplicación web, ya que, los diversos componentes que este posee evitaron que se tenga que llevar a cabo desarrollo de estilos de la aplicación.

Después de contar con la parte visual de todas las pantallas de la aplicación, se desarrolla con ayuda de la librería axios el respectivo consumo de los servicios rest que se pusieron a disposición de la aplicación con el servicio Amazon API Gateway. Con esto se pudieron obtener todos los registros de la base de datos y representarlos de forma visual en la aplicación por medio de una tabla y una gráfica. La gráfica se llevó a cabo con la librería chart.js, la cual es open source, tiene 8 tipos de gráficas y tiene compatibilidad con todos los navegadores de internet más populares del mercado.[[5]](#footnote-5) Asimismo, se utilizó la librería axios para consumir los servicios que permiten realizar la actualización de los valores de referencia mínimos y máximos del pH y la conductividad eléctrica.

Finalmente cuando todas las pantallas de la aplicación se encontraban desarrolladas y se había dado cumplimiento de todos los requerimientos funcionales que se definieron en x sección, se procedió a alojar la aplicación en el servicio previamente selecciona de Amazon Elastic BeanStalk, para lo cual fue necesario utilizar el servicio Amazon CodePipeline, el cual nos permite implementar y probar el código cada vez que se realice un cambio en este[[6]](#footnote-6), de la misma forma, permite crear un enlace con una herramienta de versionamiento online como GitHub, de tal forma que cada vez que se realice un nuevo commit a la rama con la cual se conecto al servicio de Amazon CodePipeline solo en cuestión de minutos se vea reflejado dicho cambio en la aplicación que se encuentra en línea gracias al servicio Elastic BeanStalk.

1. https://d1.awsstatic.com/whitepapers/es\_ES/aws-overview.pdf [↑](#footnote-ref-1)
2. https://d1.awsstatic.com/whitepapers/es\_ES/aws-overview.pdf [↑](#footnote-ref-2)
3. https://nodejs.org/es/about/ [↑](#footnote-ref-3)
4. https://getbootstrap.com/ [↑](#footnote-ref-4)
5. https://www.chartjs.org/ [↑](#footnote-ref-5)
6. https://d1.awsstatic.com/whitepapers/es\_ES/aws-overview.pdf [↑](#footnote-ref-6)