FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

IMPLEMENTACIÓN DE APLICACIÓN WEB QUE FACILITE EL CONTACTO ENTRE EMPLEADOR Y TRABAJADOR UTILIZANDO AWS

AUTOR

Daniel Esteban Unapanta Arias

AÑO

2020

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

IMPLEMENTACIÓN DE APLICACIÓN WEB QUE FACILITE EL CONTACTO ENTRE EMPLEADOR Y TRABAJADOR UTILIZANDO AWS

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Sistemas de Computación e Informática.

Profesor Guía

MSc. Verónica Fernanda Falconí Ausay

Autor

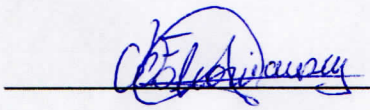
Daniel Esteban Unapanta Arias

Año

2020

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido el trabajo, Implementación de Aplicación Web que facilite el contacto entre Empleador y Trabajador utilizando AWS, a través de reuniones periódicas con el estudiante Daniel Esteban Unapanta Arias, en el semestre 202020, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".



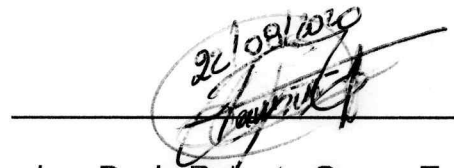
Verónica Fernanda Falconí Ausay

Magister en Ciencias de la Computación y Comercio Electrónico

C. I: 0502395270

DECLARACIÓN DEL PROFESOR CORRECTOR

"*Declaro haber revisado este trabajo, Implementación de Aplicación Web que* facilite el contacto entre Empleador y Trabajador utilizando AWS, de Daniel Esteban Unapanta Arias, en el semestre 202020, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".



Ing. Paulo Roberto Guerra Terán, MSc.

Máster en Software y Sistemas

C. I: 1002856050

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

"Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes."

Daniel Unapanta

Daniel Esteban Unapanta Arias

C. I: 1715641468

DEDICATORIA

A mi familia por su apoyo incondicional; a mi yo superior por la percepción y el entendimiento

RESUMEN

Considerando el incremento de desempleo que atraviesa actualmente el Ecuador, el presente proyecto de titulación propone una solución atrayente para solventar dicha problemática, Haciendo uso del marco de trabajo ágil Scrum y mediante la utilización de herramientas y tecnologías innovadoras de desarrollo web como lo son React y Redux para el desarrollo del frontend y servicios de Amazon Web Services que definen una arquitectura Serverless que conforma el backend. Se desarrolla una aplicación web que actúa como intermediario para conectar empleadores con trabajadores y crear contrataciones de trabajo, el usuario registrado puede alternar entre los perfiles de empleador y/o trabajador y es capaz de contratar personal calificado según sus necesidades o postularse con su curriculum según sus habilidades u oficio.

ABSTRACT

Considering the increase in unemployment that Ecuador is currently experiencing, this degree project proposes an attractive solution to solve this problem, making use of the agile Scrum framework and through the use of innovative web development tools and technologies such as React and Redux for the development of the frontend and Amazon Web Services services that define a Serverless architecture that makes up the backend. A web application is developed that acts as an intermediary to connect employers with workers and create job contracts, the registered user can alternate between employer and / or worker profiles and is able to hire qualified personnel according to their needs or apply with their curriculum according to your skills or craft.

Índice Del Contenido

Índice De Figuras

1 Capítulo I: Introducción

1.1 Antecedentes

Conforme se ha extendido el uso de la tecnología en la sociedad, se ha hecho cada vez más evidente la aparición y el auge de nuevas propuestas para solventar problemáticas o necesidades existentes, las cuales han involucrado ámbitos de todo tipo relacionadas a empleo, educación, seguridad, etc. Esto con el fin de dar respuesta a desafíos a los que se enfrenta la sociedad. Actualmente en Ecuador nos encontramos en la necesidad de solventar varias problemáticas que afectan a la población, entre ellas se encuentran: el desempleo, delincuencia, contaminación entre otras, es por ello que en el presente proyecto de titulación se presenta una solución ante la problemática relacionada al desempleo.

Actualmente en la llamada era digital se ha evidenciado el crecimiento del uso tecnológico por parte de la población, cada vez es más común el manejo de dispositivos inteligentes que permiten utilizar plataformas web y aplicaciones móviles, esto da lugar a un enorme potencial y gran cantidad de posibilidades que permitan resolver necesidades presentes en la sociedad. Es el caso historias de éxito en como Uber, Airbnb o Glovo los cuales son ejemplos de modelos de negocios exitosos que satisfacen necesidades requeridas por sus usuarios y a la vez son capaces de proveer empleo a personas que están de acuerdo en prestar sus servicios para generar ganancias.

Es fundamental tener presente ciertos datos y aspectos importantes a tener en consideración para comprender el contexto de la situación actual del país y a la problemática a la cual se pretende dar solución con el presente proyecto de titulación.

En Ecuador el hogar con acceso a internet tiene tendencia de crecimiento según los datos publicados por Tecnologías de la información y

Comunicaciones TIC, en el año 2018, el acceso a internet a nivel nacional se incrementó en 14,7 puntos a comparación del año 2012; incrementó 15,2 en el área urbana y 11.3 en el área rural, además en 2018 el 75,7% de los jóvenes entre 16 a 24 años afirmaron haber usado una computadora en el último año (INEC, 2018).

Ecuador para marzo de 2019 a nivel nacional, la tasa de participación global se ubicó en 66,8%; la tasa de desempleo fue de 4,4%; la tasa de empleo adecuado fue de 37,9%; el subempleo se ubicó en 20,3%; la tasa de otro empleo no pleno fue de 26,4% y el empleo no remunerado en 10,4% (Feijoo & del Pozo, 2019).

1.2 Alcance

La aplicación web esta implementada mediante el uso de diferentes tecnologías, en esencia se utiliza la arquitectura Serverless, una de las arquitecturas más populares para proyectos en la nube, la cual se beneficia de la facilidad para la creación y ejecución de aplicaciones y servicios sin tener que administrar Infraestructura. Se hace uso de la librería React para el desarrollo correspondiente al Frontend, adicionalmente se utiliza el contenedor de estados Redux para mejorar el comportamiento de la aplicación. En lo que respecta al backend se emplea herramientas proporcionadas por AWS: Amazon DynamoDB la cual es una base de datos noSQL capaz de devolver peticiones en microsegundos, Amazon S3 servicio para almacenar los assets y la aplicación web, Amazon Cognito servicio para el manejo de autenticación disponible en AWS, API Gateway servicio disponible para la creación y modificación de API RESTful y Funciones Lambda para la implementación y el consumo de servicios mediante programación funcional.

La aplicación web está enfocada para que el usuario registrado disponga de los perfiles correspondientes a Empleador y Trabajador por lo que puede alternar entre estos perfiles según su consideración, es decir si el usuario actúa con perfil Empleador es capaz de visualizar a los trabajadores

disponibles y realizar contrataciones según sus requerimientos, necesidades y/o consideraciones. Si el usuario actúa con perfil Trabajador es capaz de definir su curriculum para establecer su conocimiento, destrezas o habilidades de trabajo para postularse y pueda ser contratado.

En primer lugar, cuando un usuario se registra y crea su cuenta en la aplicación web; si el usuario desea tener un perfil de Trabajador es necesario establecer su curriculum, el curriculum consta de una fotografía personal actualizada y los datos e información relevante de la persona, en los que se asocia: los nombres completos, número de cédula, lugar de residencia, categoría y nombre del trabajo que ejerce, descripción de sus destrezas, especialización y/o experiencia que posee la persona, tarifa mínima y descripción personal. Es imperativo ingresar información verídica y comprobable para ya que esto da credibilidad y ayuda a un usuario empleador para tomar su decisión de contratación.

En lo que respecta al perfil Empleador simplemente es necesario que ingrese sus datos personales en el registro. El Empleador es capaz de visualizar por categorías los diferentes Servicios que puede contratar ej. (Servicio Doméstico, Servicio Cocina, Servicio Veterinario, Servicio Mecánico). Cuando el Empleador seleccionó alguna categoría, es capaz de visualizar la lista de todos los trabajadores disponibles para ser contratados, se desplegará la foto del trabajador junto con información relevante que haya incluido en su curriculum. Cuando el Empleador selecciona a un Trabajador se despliega el perfil de dicho trabajador con información más detallada del trabajo que realiza, además en el perfil del trabajador está incluida la opción de contratación. En caso de realizar la contratación se habilita para el Empleador la sección de calificación esto con el fin de validar el cumplimiento del trabajo realizado, adicionalmente se incluye la sección de review en donde es posible definir la retroalimentación que antiguos Empleadores añadieron al perfil del Trabajador contratado.

Finalmente se presenta una sección en donde el Empleador es capaz de visualizar la lista de todos los Trabajadores que ha contratado, en donde se despliega los datos más relevantes asociados a la contratación realizada

entre los cuales constan: el nombre del trabajador, trabajo realizado, fecha de contratación y calificación asignada.

1.3 Justificación

El presente proyecto de titulación exhibe una aplicación web enfocada en proporcionar una solución a la problemática del desempleo en el Ecuador. El usuario que haga uso de la aplicación web dispone de los perfiles Empleador y Trabajador y el usuario es capaz de alternar entre ellos según sus necesidades, requerimientos o criterio. La aplicación actúa como intermediario para conectar empleadores con trabajadores y crear contrataciones de trabajos, de esta manera se establece una alternativa para la generación de empleo.

1.4 Objetivo General

Implementar una aplicación web SPA utilizando servicios proporcionados por Amazon Web Services (AWS) capaz de facilitar el contacto entre Empleadores y Trabajadores de tal forma que cualquier usuario registrado en la plataforma pueda postularse para encontrar trabajo o contratar personal calificado según sus necesidades.

1.5 Objetivos Específicos

- Proporcionar una propuesta utilizando tecnologías innovadoras de desarrollo web para dar solución y afrontar la problemática del desempleo en Ecuador.

- Distribuir el desarrollo del Frontend en componentes utilizando la librería React para facilitar la construcción y mantenimiento de la aplicación web.

- Definir una arquitectura Serverless para el despliegue del backend utilizando tecnologías de Amazon Web Services.

- Desarrollar la aplicación utilizando la metodología ágil SCRUM.

2 Capítulo II: Marco Teórico

En el presente capítulo se detalla cada una de las metodologías, tecnologías y lenguajes que se hacen uso para el desarrollo del proyecto de titulación.

2.1 Metodología Scrum

La metodología Scrum es un framework iterativo e incremental de desarrollo ágil de software que se creó para ayudar a los equipos a la hora de gestionar el proceso de desarrollo. Aunque se usa principalmente en la industria de software, también se puede usar en muchas otras industrias. Scrum hace hincapié en la colaboración, el trabajo de software y la flexibilidad para adaptarse al cambio. Cuando se habla de Scrum se menciona conceptos clave tales como: transparencia, inspección y adaptación (Apiumhub, 2017).

Jeff Sutherland y Ken Schwaber crearon Scrum en los años noventa como respuesta a la forma tradicional de crear software que suponía que se puede planificar todo con precisión. Scrum se basó en la inspiración de un artículo de 1986 llamado “The New Product Development Game” (Willem-Jam, 2020).

En Scrum el proceso, el flujo de trabajo y el progreso son siempre visibles, de hecho, los equipos tienen reuniones periódicas con todos los miembros y

se recomienda la comunicación, lo que permite que el equipo se autoorganice. El equipo, generalmente este compuesto por alrededor de 7 personas, cada una con un papel diferente. La idea es que el equipo trabaje en actividades cortas llamadas “sprints”, donde la inspección y las revisiones son muy importantes. El objetivo principal es mejorar continuamente los procesos y el producto en sí (Apiumhub, 2017).

2.1.1 Product Owner

El product owner representa al cliente, siempre necesita tener una visión de lo que se necesita construir. Uno de sus principales roles es expresar y dejar claro su visión ante el equipo scrum. El producto owner es el propietario del product backlog y eso prioriza las tareas, pero esto no significa que sea el quien elija cuanto y como se hará durante el sprint (Apiumhub, 2017).

2.1.2 Scrum Master

El scrum master ayuda al producto owner y al equipo a comprender sus objetivos comunes y los asiste a la hora de planificar como lograr estos objetivos. Es asesor y entrenador de ambas partes y debe asegurarse de que el equipo alcance sus objetivos del sprint. Se recalca que el equipo al estar autoorganizado, el scrum master tiene que mantenerse neutral y no tiene realmente ninguna autoridad (Apiumhub, 2017).

2.1.3 Miembros del Equipo Scrum

El equipo esta autoorganizado y sus miembros son responsables de completar las historias de usuario que se establecieron, siempre asegurando agregar valor al producto. Una de sus tareas es dar

estimaciones para cada sprint y decidir cómo se realizará el trabajo (Apiumhub, 2017).

2.1.4 Scrum Sprint

El Scrum sprint es un ciclo de trabajo regular y repetitivo donde se termina el trabajo (work) que se determinó al comienzo del proceso, lo que lo hace listo para ser revisado. A pesar de que un scrum sprint suele ser de 30 días. se prefiere realizar iteraciones de dos semanas. Durante cada uno de los sprints, se crea un producto que se puede enviar. El producto entregado puede ser básico y eso no representa un problema debido al corto tiempo de un sprint, lo ideal es comenzar con lo más esencial. En cierto modo, esto también le da una visión del progreso al cliente. Cada vez que se inicia un nuevo sprint se trabaja y realiza iteraciones en el último que se realizó (Apiumhub, 2017).

Los sprints consisten en: el sprint palnning, el daily scrums, el trabajo de desarrollo, el sprint review y el sprint restrospective. Durante un sprint no se deben realizar cambios que pongan en peligro al sprint goal y el alcance puede aclararse y renegociarse entre el propietario del producto y el equipo de desarrollo a medida que se aprenda más (Schwaber & Sutherland, 2013).

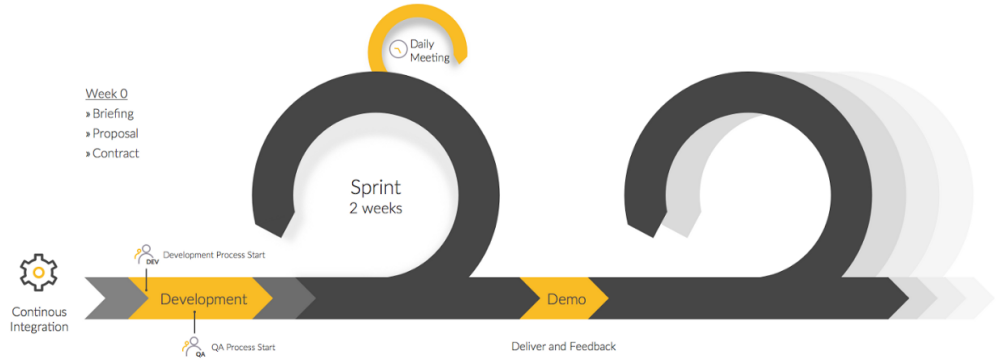


Figura 1. Scrum Sprint

Tomado de (Apiumhub, 2017)

2.1.5 Sprint Planing

Todos los spints deben comenzar con una reunión donde el equipo discuta y planifique/organice el sprint. Primero se comienza estableciendo metas y determinando cuáles serán los entregables para el sprint. El equipo identifica las historias de usuario que se trasladaran desde el product backlog al backlog del sprint (lista de tareas para el sprint). Básicamente se establece qué se hará y como se hará. Es importante recalcar que el trabajo no debe ser agregado al sprint una vez ha comenzado. Además, si no se ha hecho algo al final del sprint, solo se debe pasarlo al trabajo atrasado y priorizarlo (Apiumhub, 2017).

2.1.6 Daily Scrum

El día inicia con la reunión diaria de scrum. Una breve reunión donde los miembros del equipo hacen un chequeo para ayudar a resolver problemas. Los miembros hablan y ven cómo va todo, que se ha hecho, que se hará durante este día y si hay algún problema encontrado. También se lo conoce como “standup” (Apiumhub, 2017).

2.1.7 Scrum Sprint Review

La revisión del sprint marca el final “público” del sprint. Todos los interesados deben estar presentes durante la reunión y el equipo tiene la oportunidad de hablar sobre las historias de los usuarios que no se pudieron completar (si las hay) y luego pueden mostrar el trabajo realizado. Desde el otro lado el producto owner puede ver las mejoras

realizadas al producto. Durante esta fase, la retroalimentación es muy importante (Apiumhub, 2017).

2.2 Tecnologías, Lenguajes y Herramientas de Desarrollo

2.2.1 Single Page Aplication (SPA)

Single Page Aplication (SPA) es el nombre que toma una aplicación que carga una única página en HTML junto con todos sus assets requeridos para que la aplicación se ejecute, la principal característica de una SPA son las interacciones con la página o paginas posteriores no requieren un viaje de ida y vuelta al servidor, lo cual significa que la pagina no vuelve a cargar, (React, 2019) esto se debe a que la aplicación se ejecuta en el lado del navegador como una sola página web que va cargando o renderizando partes del contenido de la interfaz por medio de Javascript basándose en las interacciones del usuario, algunos ejemplos de SPA son: Gmail, Google Maps, Facebook, Instagram o Github.

Las webs de tipo Single Page Aplication permiten al usuario permanecer en un único sitio web lo cual genera una gran experiencia donde se presenta el contenido de manera simple, fácil y práctica (Neoteric, 2016).

Algunas ventajas de utilizar Single Page Aplication son las siguientes:

- Las webs de tipo Single Page Aplication se comportan más rápido ya que la mayoría de los recursos (HTML, CSS, Scripts) solo se cargan una vez a lo largo de la vida útil de la aplicación. Solo los datos son transmitidos de ida y vuelta.

- El desarrollo se simplifica, ya que no es necesario escribir código para renderizar páginas en el servidor. Además, incluir lógica propia del servidor hacia el lado del cliente para realizar decisiones más rápidamente.

- Las aplicaciones de tipo Single Page Aplication son fáciles de depurar desde Chrome ya que se pueden monitorear las operaciones de la red e investigar los elementos de la página y los datos asociados a ella.

- Las webs de tipo Single Page Aplication pueden almacenar en cache cualquier almacenamiento local de manera efectiva.

2.2.2 Arquitectura Serverless

La Arquitectura Serverless hace referencia al modelo de ejecución de Cloud Computing en donde el proveedor Cloud administra dinámicamente la distribución y aprovisionamiento de los servidores (Bashir, 2018). También se puede definir como aplicaciones que dependen significativamente de servicios de terceros (Backend as a Service o "BaaS") o código personalizado que se ejecuta en contenedores efímeros (Function as a Service o "FaaS") (Techlabs, 2017).

- **Backend as a Service (BaaS):** Es el término al backend de tipo Serverless, un backend de alta disponibilidad que se puede configurar con casi cualquier configuración y puede escalar casi indefinidamente (Dao, 2018). Entre los servicios que se ofrecen generalmente para las aplicaciones están: administración de estado, almacenamiento de datos, autenticación, notificaciones, entre otros (Nawaz, 2017).

- **Function as a Service (FaaS):** Es el término que se refiere al producto Serverless que almacena una parte de la lógica del negocio, Fass es muy conveniente para arquitecturas dirigidas por eventos (Dao, 2018). Faas implica que el desarrollador escribe piezas de código en el lado del servidor o "funciones" y se las carga en una plataforma Serverless en donde estas funciones se activarán y ejecutarán cuando se lo requiera y además solo se paga por el tiempo de ejecución (Nawaz, 2017).

Entre las principales ventajas de usar una arquitectura Serverless contra una tradicional están:

- **Precio:** Al utilizar una arquitectura Serverless se evidencia la reducción del costo a pagar, el modelo de costo está basado en ejecución, es decir que se cobra por el número de ejecuciones realizadas, se asigna un cierto número de segundos de uso que varía con la cantidad de memoria que se requiere, de esta manera funciones con ejecución más corta son más adaptables a este modelo con un tiempo máximo de ejecución de 300 segundos para la mayoría de proveedores Cloud (Bashir, 2018).

- **Escalamiento:** Una de las principales ventajas de la arquitectura Serverless es el proceso de escalado ya que es automático y continuo (Bashir, 2018). Si el código requiere escalar, la plataforma realiza copias de la función para manejar la carga, es decir en horas pico de uso de la aplicación, la plataforma se encarga de escalar con decenas de instancias automáticamente, mientras que si la demanda a la aplicación disminuye revierte el escalamiento realizado (Nawaz, 2017).

- **Productividad:** La arquitectura Serverless es especialmente útil para equipos de desarrollo que requieran de un rápido desarrollo, prototipado e iteración. El desarrollo es más veloz debido a que no se requieren ninguna dependencia en DevOps. El código también suele ser de un solo subproceso, lo que facilita la depuración (Nawaz, 2017).

Actualmente existen gran cantidad de proveedores Cloud que facilitan la utilización de una arquitectura Serverless entre los principales proveedores están:

- AWS Lambda
- Google Cloud Functions
- Azure Functions

- IBM OpenWhisk
- Oracle Fn Project
- Kubeless
- Alibaba Function Compute

(Chowman, 2018)

Figura 2. Principales Proveedores Cloud

Tomado de (Bashir, 2018)

Existen múltiples soluciones de arquitectura Serverless que pueden variar según los requerimientos y necesidades de los clientes, es posible crear cualquier tipo de aplicación o servicio backend con una arquitectura sin servidor, a continuación, se presenta un caso práctico común de arquitectura Serverless en AWS que utiliza varios de los servicios disponibles en AWS como son: AWS Lambda, Amazon API Gateway, Amazon S3, Amazon DynamoDB y Amazon Cognito. (Amazon Web Services, 2020).

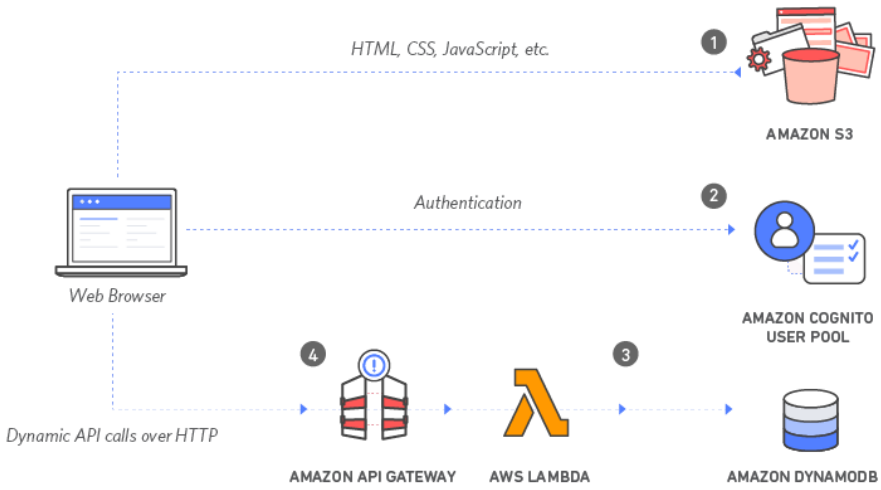


Figura 3. Arquitectura Serverless en AWS

Tomado de (Amazon Web Services, 2020)

2.2.3 Lenguajes de Programación

2.2.3.1 JavaScript ECMAScript 6 (ES6)

Las siglas ES6 se refiere a las más recientes versiones del estándar ECMAScript Language Specification del cual el lenguaje Javascript es una implementación. En esta nueva versión se han incluido nuevas características como son: funciones flecha, clases, platillas de cadenas de texto, módulos, promesas, etc (React, 2019).

2.2.3.2 JSX

JSX es una extensión de la sintaxis de Javascript, se recomienda utilizar este lenguaje con React para describir el diseño de la interfaz de usuario (React, 2019). Al utilizar JSX con React después de compilarse, las expresiones JSX se transforman en llamadas a funciones regulares Javascript y se evalúan en objetos Javascript.

2.2.3.3 JSS

JSS es una herramienta de creación de CSS que permite usar JavaScript para describir estilos de manera declarativa, libre de conflictos y reutilizable. Se puede compilar en el navegador, del lado del servidor o en el momento de la compilación de Node (JSS, s.f.).

2.2.4 Tecnologías para Desarrollo de Frontend

2.2.4.1 React

React es una biblioteca de Javascript para desarrollar interfaces de usuario para webs de tipo Single Page Aplication, dividiendo la interfaz de usuario en componentes componibles (Sahin, 2019).

React cumple con tres principios principales los cuales son:

- **Declarativo:** React simplifica la creación de interfaces de usuario interactivas, el diseñar vistas simples para cada estado de la aplicación está en manos del desarrollador y React se encarga de renderizar y actualizar eficientemente los componentes correctos cuando cambien sus datos. Las vistas declarativas hacen que el código sea más predecible, más sencillo de entender y más fácil de depurar.
- **Basado en Componentes:** Se basa en crear componentes encapsulados que administren su propio estado y luego unirlos para crear interfaces de usuario complejas. Dado que la lógica de componentes está escrita en Javascript, se puede pasar fácilmente datos enriquecidos a través de la aplicación y mantener el estado fuera del DOM.
- **Aprender una Vez, Escribirlo en Cualquier Lugar:** No se hacen suposiciones sobre el resto de la pila tecnológica, por lo que se puede desarrollar nuevas funciones en React sin reescribir el código existente. React también puede renderizar

en el servidor usando Node y potenciar aplicaciones móviles usando React Native.

(Facebook, 2020)

2.2.4.2 Redux

Redux es un contenedor predecible del estado de aplicaciones Javascript. Es una librería de apenas 2kB que permite controlar el estado de aplicaciones web de una manera más sencilla, permite escribir aplicaciones que se comportan de manera consistente, que corren en distintos ambientes (cliente, servidor y nativo) y que sean fáciles de probar.

Redux se basa en tres principios fundamentales listados a continuación:

- **Única fuente de la verdad**: Este principio hace referencia a que el estado de toda la aplicación este contenido en un único Stotre, esto permite crear fácilmente aplicaciones universales, debido a que el estado del servidor puede ser serializado y enviado al cliente sin esfuerzos extras, de igual manera al tener un único Store, esto hace que sea mucho más fácil depurar la aplicación.

- **El estado es de solo lectura:** El segundo principio establece que la única manera de modificar el estado es emitiendo una acción que indique que cambió, con esto se asegura ninguna vista o callback de red modifiquen el estado directamente, por el contrario, envían una intención de modificarlo, esto a su vez al estar centralizado permite que los cambios se ejecuten uno a la vez en un orden estricto.

- **Los cambios se realizan con funciones puras:** Para controlar como las modificaciones hacia el Store son realizadas por las

acciones se utilizan reducers puros. Los reducers son funciones puras que toman el estado anterior y una acción, y entregan el nuevo estado. Se puede tener un solo reducer para toda la aplicación o dividirlo en varias funciones que manejen partes específicas del Store.

A continuación, se detallan los conceptos que utiliza Redux para funcionar en aplicaciones web:

- **Acciones:** Las acciones son las encargadas de enviar datos desde la aplicación al Store, son consideradas POJOs (Plain Old JavaScript Objects) que posee al menos una propiedad que indica el tipo de acción y si es necesario otras propiedades que permitan efectuar la acción se las envía por medio de *store.dispatch().* (Xalambrí, 2018).

- **Reducers:** Los reducers son los encargados de definir el cambio de estado de la aplicación, al reducer se lo define como una función pura que toma el estado anterior y una acción y se encarga de devolver el nuevo estado.
  Al ser una función pura implica que al pasar los mismos parámetros debería devolver exactamente el mismo resultado, es por ello que ciertas acciones no deben estar definidos dentro del reducir como es:

  - Modificar directamente sus argumentos
  - Realizar tareas con efectos secundarios como llamadas a API o transiciones de rutas
  - Ejecutar funciones no puras

- **Store:** En Redux el Store es el objeto que posee ciertas responsabilidades listadas a continuación:

    - Almacena el estado de la aplicación
    - Permite el acceso al estado de la aplicación por medio de *store.getState().*
    - Permite que el estado se actualice por medio de *store.dispatch(action).*
    - Registra los listeners por medio de *store.subscribe(listener).*

La arquitectura Redux se basa en un flujo de datos estrictamente unidireccional, todos los datos de la aplicación siguen el mismo patrón de ciclo de duración, haciendo que la lógica de la aplicación sea más predecible y fácil de entender.

(Silva, González Reina, Hereñú, & Teruel, 2019)

A continuación, se presenta el flujo de datos de Redux

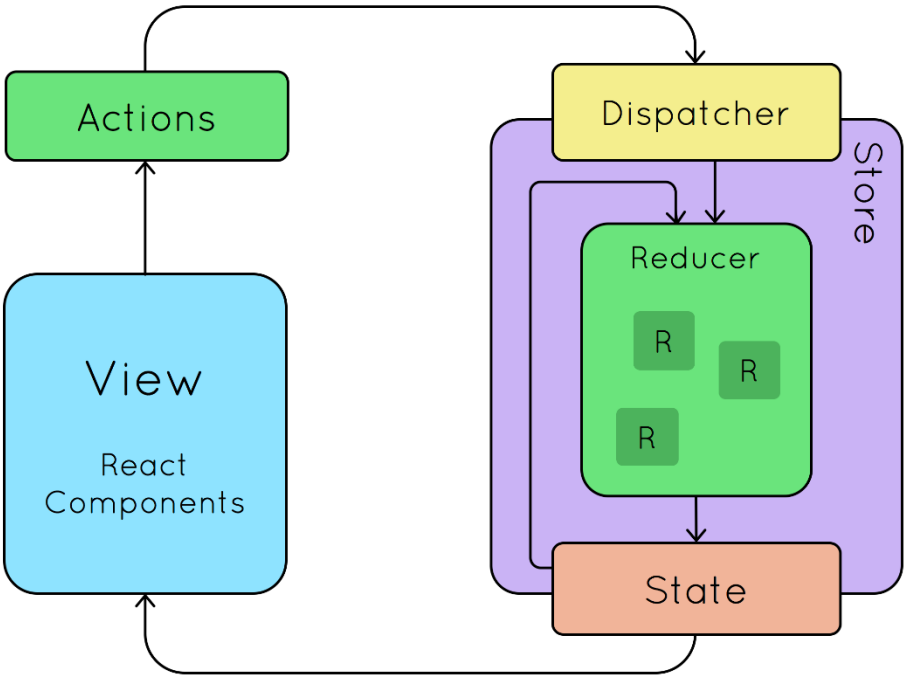


Figura 4. Flujo de Trabajo de Redux

Tomado de (Van den Bergh, 2017)

2.2.4.3 Material-UI

Material-UI es un framework gratis de componentes, fuentes e iconos que se pone a disposición de desarrolladores para ofrecer un desarrollo de aplicaciones web rápido y sencillo utilizando la librería React. Material-UI está disponible a través de paquetes npm o mediante la utilización de CDN. Los componentes de la interfaz de usuario de Material-UI funcionan de forma aislada, son autosuficientes y solo inyectan los estilos que necesitan para desplegar (Material-UI, s.f.).

2.2.4.4 Babel

Babel es una cadena de herramientas que se utiliza principalmente para convertir el código ECMAScript 2015+ en una versión de

Javascript compatible con versiones anteriores en navegadores o entornos actuales y anteriores. Entre las principales funciones que puede realizar Babel están:

- Transformar la sintaxis.
- Realizar funciones de polyfill que faltan en un entorno de destino.
- Transformaciones de código fuente (codemods).
- Transformación de sintaxis JSX y React a través del paquete babel-sublime.

(Babel, s.f.)

2.2.4.5 Webpack

Webpack es un generador de módulos para aplicaciones JavaScript modernas. Cuando webpack procesa una aplicación, internamente genera un grafo de dependencias en donde mapea cada módulo que un proyecto necesita y genera uno o más paquetes (webpack, s.f.). Webpack es una herramienta que utiliza una configuración para explicarle al “constructor” como cargar cosas específicas, de esta manera la aplicación conoce que archivos necesita, en qué orden los necesita y de qué depende cada pieza, luego webpack creará paquetes tan pocos como sea posible y tan optimizados como sea posible, que se los incluirá como script en la aplicación (Ciel, 2017) .

Entre las principales operaciones que puede realizar webpack estan:

- Ayuda a agrupar los recursos de la aplicación.
- Vigila los cambios y vuelve a ejecutar las tareas.
- Puede ejecutar la transpilación de Babel a ES5 lo que permite utilizar últimas características de JavaScript sin preocuparse por compatibilidad del navegador.
- Convertir imágenes en línea a URI de datos.
- Puede ejecutar un servidor web de desarrollo.

- Puede dividir los archivos de salida en varios archivos para evitar tener que cargar un gran archivo JS en la primera carga de la página.
- Puede realizar “tree shacking” (remover código nunca usado).

(Copes, 2018)

2.2.5 Herramientas Cloud de AWS para Implementación de Arquitectura Serverless y Desarrollo de Backend

2.2.5.1 AWS Amplify

AWS Amplify es una plataforma de desarrollo para la creación de aplicaciones móviles y web seguras que sean escalables. Con AWS Amplify se facilita la autenticación de usuarios, almacenamiento seguro de datos y metadatos del usuario, autorización de acceso selectivo a los datos, integración con aprendizaje automático, análisis de métricas y ejecución de código del lado del servidor. Cabe recalcar que las bibliotecas de Amplify y CLI, parte del framework Amplify, es de código abierto y ofrece una interfaz conectable que permite crear y personalizar complementos propios (Amazon Web Services, 2020).

Figura 5. Logo Amplify

Tomado de (Amazon Web Services, 2020)

El framework Aws Amplify se compone de las siguientes herramientas y servicios:

- **AWS Amplify CLI:** Es una cadena de herramientas que incluye un robusto conjunto de características para simplificar el desarrollo de aplicaciones móviles y web. CLI usa AWS CloudFormation y pilas anidadas que permite agregar o modificar configuraciones localmente antes de ser enviadas a AWS (Github, s.f.).
- **API as a Service:** Simplemente es necesario configurar el modelo de datos utilizando GraphQL Schema Definition Language (SDL), y Amplify se encargará de crear y administrar todos los recursos. También ofrece subscripciones utilizando WebSockets para añadir características de tiempo real (Bougère, 2019).
- **Manejo de Servicio de Hosting:** En ambientes configurados con Amplify es posible construir la aplicación frontend, que se desplegara en S3 y configura la distribución de CloudFront (Bougère, 2019).
- **Librería UI:** Amplify provee de componentes UI para React, React Native, Angular, Ionic y Vue (Bougère, 2019).

2.2.5.2 Amazon Simple Storage (Amazon S3)

Amazon Simple Storage, mejor conocido como Amazon S3, es una solución altamente escalable, rápida y duradera para el almacenamiento a nivel de objetos de cualquier tipo de datos (Hernandez, 2019). Amazon S3 posee de una interfaz de servicios web sencilla que es usada para almacenar y recuperar cualquier

cantidad de datos, en cualquier momento, desde cualquier parte de la web. Ofrece a cualquier desarrollador la capacidad de acceder a la misma infraestructura de almacenamiento de datos económica, altamente escalable, fiable, segura y rápida que utiliza Amazon para mantener su propia red global de sitios web. Este servicio tiene como fin maximizar los beneficios del escalado y proveerlos a los desarrolladores (Amazon Web Services, 2020).

Existen varias características al momento de la utilización de Amazon Web Services que se destacan de otros servicios de almacenamiento de datos entre las que se encuentran (Hernandez, 2019):

- Transferencia de datos hacia S3 por medio de AWS S3 API, lo cual facilita la carga y automatización de copias de seguridad.
- Capacidad de escribir, leer y borrar objetos desde 1 byte a 5 terabytes.
- Numero de objetos ilimitado.
- Mecanismo de autenticación para permitir autenticación y denegar el acceso no autorizado de usuarios externos.
- Interfaces API REST y SOAP.
- Simplicidad en gestión de datos al segregarlos por “buckets”, monitorear el acceso y controlar los ciclos de vida de los datos.

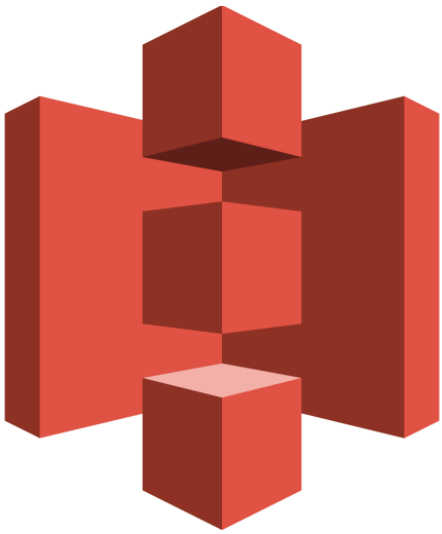


Figura 6. Logo Amazon S3

Tomado de (BrandEPS, 2020)

2.2.5.3 Amazon Cognito

Amazon Cognito es un producto de Amazon Web Services que controla la autenticación de usuarios y el acceso para aplicaciones web y móviles en dispositivos conectados a internet. El servicio guarda y sincroniza los datos del usuario final, lo que permite a un desarrollador centrarse en escribir código en lugar de desarrollar y administrar la infraestructura de back-end. Esto puede acelerar el proceso de desarrollo de aplicaciones web y móviles (TechTarget, 2020).

Amazon Coginto proporciona autenticación, autorización y administración de usuarios, los usuarios tienen la posibilidad de iniciar sesión ya sea con un nombre de usuario y contraseña o a través de un tercero como Google, Facebook, Amazon o Apple (Amazon Web Services, 2020).

Los componentes principales de Amazon Cognito son:

- **Grupos de Usuario:** Son directorios de usuario que proporcionan a los usuarios de las aplicaciones opciones para inscribirse e iniciar sesión.
- **Grupos de Identidades:** Permiten conceder a los usuarios acceso a otros servicios de AWS.

Figura 7. Logo Amazon Cognito

Tomado de (Amazon Web Services, 2020)

2.2.5.4 Amazon API Gateway

Amazon API Gateway es un servicio de Amazon Web Services para la creación, publicación, mantenimiento, monitoreo y seguridad de REST, HTTP y WebSockets APIs a cualquier escala. Los desarrolladores a través de API Gateway pueden crear APIs que puedan acceder a AWS u otros servicios web, como también para uso de aplicaciones de cliente propias u ofrecer la API a desarrolladores externos. API Gateway proporciona una experiencia de desarrollo integrada y coherente para la creación de aplicaciones Serverless. API Gateway maneja todaslas tareas involucradas en manejar y procesar hasta cientos de miles de llamadas API simultaneas. Estas tareas incluyen la administración del tráfico, el control de la autorización y el acceso, la monitorización y la administración de versiones de la API. (Amazon Web Services, 2020).

En lo que respecta a la creación de API Restful a través de API Gateway se toma en cuenta:

- Se basa en HTTP.
- Habilitan la comunicación entre cliente y servidor sin estado

- Implementan métodos HTTP estándar como GET, POST, PUT, PATCH y DELETE

Figura 8. Logo Api Gateway

Tomado de (KindPNG, 2020)

2.2.5.5 AWS Lambda

AWS Lambda en un servicio informático que permite ejecutar código sin aprovisionar ni administrar servidores, el código es ejecutado en respuesta a eventos y automáticamente gestiona los recursos informáticos subyacentes que son requeridos. Es posible utilizar AWS Lambda para extender otros servicios de AWS con lógica personalizada o crear servicios backend que operan en escala, rendimiento y seguridad de AWS (Sri, 2020). AWS Lambda ejecuta el código en una infraestructura informática de alta disponibilidad y ejecuta la administración integral de los recursos informáticos, incluido el mantenimiento del servidor, del sistema operativo, el aprovisionamiento de capacidad, escalado automático, monitorización

y los registros. Simplemente se debe suministrar el código en uno de los lenguajes admitidos por AWS Lambda (Amazon Web Services, 2020).

El código que se ejecuta en AWS Lambda se denomina “función Lambda”, una vez haber definido el código, siempre estará disponible para ser ejecutado tan pronto se lo active. Las funciones Lambda no poseen estado, no tienen afinidad por la infraestructura subyacente, por lo que Lambda puede iniciar rápidamente tantas copias de la función como sea necesario para aumentar la tasa de eventos recibidos (Sri, 2020).

Existen varias ventajas de utilizar AWS Lambda en especial en arquitecturas Serverless tales como:

- **Escalado Continuo:** El código se ejecuta en paralelo y procesa cada desencadenante individualmente, escalando con precisión con el tamaño de carga de trabajo.
- **Segundos de Medición:** En AWS Lambda se cobra por cada 100 ms que se ejecuta el código y la cantidad de veces que se activa, por lo que se paga por el tiempo de procesamiento consumido.
- **Rendimiento Consistente:** Se puede optimizar el tiempo de ejecución del código al elegir el tamaño de memoria adecuado para la función. También es posible habilitar la concurrencia aprovisionada para mantener funciones inicializadas y preparadas para responder en milisegundos de dos dígitos (Sri, 2020).

Figura 9. Logo AWS Lambda

Tomado de (Services, 2020)

2.2.5.6 Amazon DynamoDB

Amazon DynamoDB es un servicio de base de datos NoSQL completamente administrado que proporciona un rendimiento rápido y predecible con datos integrales. Permite descargar las cargas administrativas de operar y escalar una base de datos distribuida sin preocuparse del aprovisionamiento, instalación y configuración de hardware, tareas de replicación o escalado de clústeres (Shevindi, 2019).

Entre las principales características de Amazon DynamoDB se encuentran:

- **Escalado:** DynamoDB está diseñado para un rendimiento continuo y escalado de almacenamiento.
- **Rendimiento Predecible:** El servicio se ejecuta en discos de estado sólido y está diseñado para mantener latencias rápidas y consistentes en cualquier escala.

- **Fácil Administración:** DynamoDB es un servicio totalmente administrado en donde el usuario se encarga de crear las tablas de base de datos y el servicio se encarga del apartado técnico.
- **Tolerancia ante Fallos:** DynamoDB replica de forma automática y síncrona los datos en múltiples zonas de disponibilidad en una región para una alta disponibilidad.
- **Flexible:** DynamoDB no posee un esquema fijo, cada elemento de datos puede tener un numero diferente de atributos. Multiples tipos de datos agregan riqueza al modelo de datos.
- **Seguro:** DynamoDB es seguro y utiliza métodos criptográficos probados para autenticar a los usuarios y evitar el acceso no autorizado a los datos. También se integra con AWS Identity y Access Management para su control de acceso.
  (Cynix, 2019)

Figura 10. Logo Amazon DynamoDB

Tomado de (Lai, 2019)

3 Capitulo III: Planificación de Metodología Scrum, Propuesta y Diseño de la Aplicación

En este capítulo se presenta la definición del Product Backlog del proyecto, las historias de usuario y el Diagrama de la aplicación.

3.1 Product Backlog

A continuación, se presenta el Product Backlog correspondiente al proyecto. Para la estimación se hace uso de Planing Poker, una técnica empleada para calcular el esfuerzo de cada ítem definido. Se hace uso de una baraja de cartas con una distribución de números similar a la secuencia Fibonacci, en donde la medida de estimación corresponderá a puntos de historia o PH que podrán tomar valores del 1 al 13 (1, 2, 3, 5, 8 o 13), se toma en cuenta que cada punto de estimación corresponde a 5 horas de trabajo aproximadamente de esta manera es posible definir el esfuerzo esperado para cada historia de usuario que conformará el proyecto.

Tabla 1.

Product Backlog del Proyecto

ID	Como un	Deseeo..	Para..	Notas	Prioridad	Estimación
1	Empleador y Trabajador	crear una nueva cuenta en la plataforma	iniciar sesión y tener acceso a las funcionalidades de la aplicación.	Validar con Cognito los tokens correspondientes	Requerida	5PH
2	Trabajador	acceder a la página	navegar entre los servicios de la aplicación	El usuario debe haber iniciado sesión	Requerida	5PH

		principal de la aplicación				
3	Trabajador	crear un curriculum personalizado	dar a conocer las aptitudes, capacidades y experiencias del trabajo que desempeño	la información debe almacenarse ya sea en DynamoDB o S3 según consideraciones a tomar.	Requerida	13PH
4	Empleador	visualizar el despliegue de todos los empleados disponibles según la categoría seleccionada	decidir por el trabajador que mejor conviene para el trabajo	El despliegue de los trabajadores debe presentar información relevante sobre el curriculum del trabajador.	Requerida	5PH
5	Empleador	contratar a determinado empleado	Obtener los datos contacto del trabajador y establecer comunicación	Se debe crear un componente que vincule la acción de contratar entre un empleado y un trabajador	Requerida	13PH
6	Empleador	Desplegar lista de trabajadores contratados	Verificar el registro de las contrataciones realizadas	Obtener datos de las tablas de DynamoDB para presentar la información de contratación	Requerida	5PH

3.2 Historias de Usuario

A continuación, se definen las historias de usuario correspondiente a cada sprint que constituye el proyecto

- **ID**: identificador de la historia de usuario, único para la funcionalidad o trabajo.
- **Usuario:** de dónde se obtuvo.
- **Nombre de historia:** nombre descriptivo de la historia de usuario.
- **Prioridad en negocio:** sistema de priorización que permite determinar el orden en el que las historias de usuario deben de ser implementadas.
- **Riesgo en desarrollo:** riesgo técnico o funcional asociado a la implementación de la historia de usuario.
- **Estimación:** estimación del esfuerzo necesario ideal de desarrollo de la historia de usuario.
- **Sprint asignado:** puede ser útil para organización del propietario del producto incluir el número de sprint en el que previsiblemente se vaya a realizar la historia.
- **Responsable:** en casos de sugerir la persona que pueda implementar la historia de usuario.
- **Descripción:** descripción sintetizada de la historia de usuario. El estilo puede ser libre, según convenga, debe responder a tres preguntas: ¿Quién se beneficia? ¿Qué se quiere? y ¿Cuál es el beneficio?
- **Criterios de aceptación:** pruebas de aceptación consensuadas con el cliente o usuario.

Tabla 2.

Historia de Usuario Registro de Nuevo Usuario

Historia de usuario	
ID: 01	**Usuario:** Daniel Unapanta
Nombre de historia: Registro de Nuevo Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 5PH	**Sprint asignado:** 1
Responsable: Daniel Unapanta	
Descripción: Como empleador y trabajador, deseo crear una nueva cuenta en la plataforma, para iniciar sesión en la aplicación web.	
Criterios de aceptación: El usuario es capaz de navegar entre las secciones de registro, verificación de código o inicio de sesión • Se debe proporcionar la sección de registro con los campos correspondientes a nombre de usuario, email, contraseña y número de contacto. • Se debe enviar un código de verificación al correo proporcionado por el usuario. • Se debe proporcionar un campo de verificación para ingresar el código generado y verificar la cuenta creada. • Se debe implementar el inicio de sesión con los campos de nombre de usuario y contraseña.	

Tabla 3.

Historia de Usuario Home Page

Historia de usuario	
ID: 02	**Usuario:** Daniel Unapanta
Nombre de historia: Home Page	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 5PH	**Sprint asignado:** 2
Responsable: Daniel Unapanta	
Descripción: Como empleador y trabajador, deseo acceder a la página principal de la aplicación, para conocer y navegar entre los diferentes servicios que proporciona la aplicación web.	

Criterios de aceptación: El usuario debe ser capaz de acceder a la página principal de la aplicación web después de haber iniciado sesión. • Se debe proporcionar una barra de navegación para facilidad de uso y acceso a los servicios de definición de curriculum, acceso a categorías, despliegue de contrataciones y cierre de sesión. • Se debe definir el contenido principal referente a los fines de la aplicación web. • Se debe definir el pie de página.

Tabla 4.

Historia de Usuario Creación De Currículum Personalizado para Trabajador

Historia de usuario	
ID: 03	**Usuario:** Daniel Unapanta
Nombre de historia: Creación De Currículum Personalizado para Trabajador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 13PH	**Sprint asignado:** 3
Responsable: Daniel Unapanta	
Descripción: Como trabajador, deseo crear un curriculum personalizado, para dar a conocer las aptitudes, capacidades y experiencias del trabajo que realizo.	
Criterios de aceptación: El usuario debe ser capaz de acceder a la sección de creación de curriculum en caso de querer postularse como trabajador • Se debe proporcionar la sección de fotografía de curriculum en donde el usuario puede subir una fotografía personal en formato jpg o png. • Se debe proporcionar la sección de definición de trabajo en donde el usuario es capaz de ingresar los datos de su curriculum, entre los campos a ingresar constan: nombres, apellidos, cédula, teléfono, categoría de trabajo, nombre trabajo, tarifa, ciudad, país, código postal, acerca de mi y experiencia laboral.	

Tabla 5.

Historia de Usuario Despliegue de Trabajadores Disponibles Según Categoría

Historia de usuario	
ID: 04	**Usuario:** Daniel Unapanta
Nombre de historia: Despliegue de Trabajadores Disponibles Según Categoría	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 5PH	**Sprint asignado:** 4
Responsable: Daniel Unapanta	
Descripción: Como empleador, deseo visualizar el despliegue de todos los trabajadores disponibles según la categoría seleccionada, para decidir por el trabajador que mejor se adapta a las necesidades del trabajo.	
Criterios de aceptación: El usuario debe ser capaz de acceder a la lista de trabajadores disponibles para ser contratados • Se debe proporcionar un menú desplegable con el nombre de las categorías de trabajos disponibles en donde se filtran los trabajadores por categoría. • Se debe proporcionar la sección despliegue de trabajadores, para cada trabajador se debe incluir la información de: trabajo, categoría, fotografía personal, nombre, tarifa, acerca de mi y un botón de visualización de perfil.	

Tabla 6.

Historia de Usuario Visualización de perfil de Trabajador y Contratación

Historia de usuario	
ID: 05	**Usuario:** Daniel Unapanta
Nombre de historia: Visualización de perfil de Trabajador y Contratación	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 13PH	**Sprint asignado:** 5
Responsable: Daniel Unapanta	
Descripción: Como empleador, deseo visualizar el curriculum del trabajador seleccionado de la lista, para validar los servicios proporcionados y verificar si se adapta a mis necesidades de trabajo y en tal caso realizar la contratación.	

Criterios de aceptación: El usuario debe ser capaz de visualizar el perfil de cualquier trabajador para validar su información y poder ser contratado. • Se debe proporcionar la sección de perfil de trabajador en donde se despliega información relevante de su curriculum: fotografía personal, nombre, tarifa, ciudad, país, acerca de mí y experiencia. • Se debe definir un botón para la realización de una contratación • Se debe proporcionar un modal que incluya información de la contratación realizada en donde se debe especificar el nombre del trabajador contratado, el trabajo, su correo electrónico y su número de contacto. • Se debe proporcionar la sección de calificación que incluya la puntuación con estrellas y el review de la contratación que fue realizada.

Tabla 7.

Historia de Usuario Despliegue de Lista de Todos los Trabajadores Contratados

Historia de usuario	
ID: 06	**Usuario:** Daniel Unapanta
Nombre de historia: Despliegue de Lista de Todos los Trabajadores Contratados	
Prioridad en negocio: Media	**Riesgo en desarrollo:** N/A
Estimación: 5PH	**Sprint asignado:** 6
Responsable: Daniel Unapanta	
Descripción: Como empleador, deseo desplegar la lista de todos los trabajadores que he contratado, para verificar el registro de contrataciones realizadas.	
Criterios de aceptación: El usuario debe ser capaz de visualizar el listado de las contrataciones realizadas. • Se debe proporcionar la sección de trabajadores contratados la cual consta del listado correspondiente a cada contratación, se de incluir los campos de trabajador, categoría, trabajo, fecha de contratación y calificación. • Se debe incluir un botón capaz de dirigir al usuario a la sección de calificación en caso de no haber calificado a un trabajador.	

3.3 Diagrama de la Aplicación

A continuación, se presenta el diagrama de la aplicación

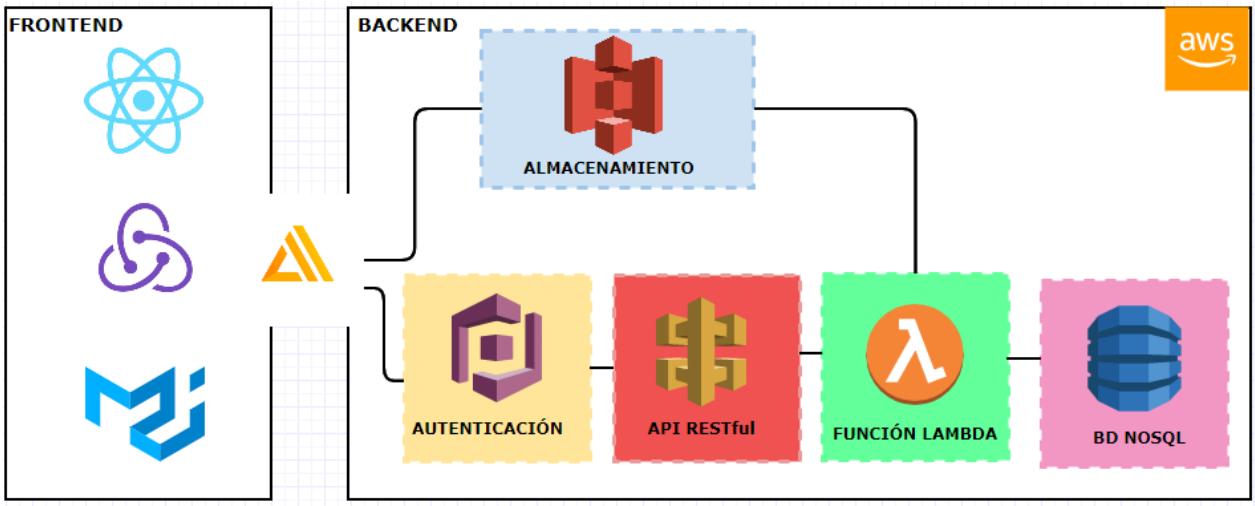


Figura 11. Diagrama Aplicación

4 Capitulo IV: Desarrollo

4.1 Primer Sprint

A continuación, se redacta el procedimiento que se llevó a cabo para la culminación del primer sprint.

4.1.1 Sprint Planning

En lo que respecta a la reunión del sprint planning del primer sprint, se determina la organización y planificación del mismo, al ser el primer sprint se establece enfocarse en determinar cuáles serán las metas y los primeros entregables de la aplicación web, en este caso se establece trabajar con la autenticación del usuario en la aplicación por lo que se

definen tres nuevas historias de usuario a realizar en el backlog del sprint a partir del producto backlog correspondiente a la autenticación, estas serán: Registro, Inicio de Sesión y Verificación de Código. En la reunión se define que componentes se implementarán y como se implementarán, en este caso se define utilizar React Hooks para estos componentes.

4.1.2 Historias de Usuario del Sprint Backlog del Primer Sprint

Tabla 8.

Historia de Usuario Registro de Nuevo Usuario

Historia de usuario	
ID: AUTH_01	**Usuario:** Daniel Unapanta
Nombre de historia: Registro de Nuevo Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 2PH	**Sprint asignado:** 1
Responsable: Daniel Unapanta	
Descripción: Como empleador y trabajador, deseo crear una nueva cuenta en la plataforma, para obtener un token de autenticación del user pool de cognito.	
Criterios de aceptación: Se debe proporcionar la sección de registro con los campos correspondientes a nombre de usuario, email, contraseña y número de contacto.	

Tabla 9.

Historia de Usuario Código de Verificación

Historia de usuario	
ID: AUTH_02	**Usuario:** Daniel Unapanta
Nombre de historia: Código de Verificación	
Prioridad en negocio: Media	**Riesgo en desarrollo:** N/A

Estimación: 1PH	**Sprint asignado:** 1
Responsable: Daniel Unapanta	
Descripción: Como empleador y trabajador, deseo obtener un código de verificación al crear una cuenta para tener seguridad.	
Criterios de aceptación: • Se debe enviar un código de verificación al correo proporcionado por el usuario. • Se debe proporcionar un campo de verificación para ingresar el código generado y verificar la cuenta creada.	

Tabla 10.

Historia de Usuario Inicio de Sesión

Historia de usuario	
ID: AUTH_03	**Usuario:** Daniel Unapanta
Nombre de historia: Inicio de Sesión	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 2PH	**Sprint asignado:** 1
Responsable: Daniel Unapanta	
Descripción: Como empleador y trabajador, deseo iniciar sesión con la cuenta previamente creada para acceder a las funcionalidades de la aplicación.	
Criterios de aceptación: Se debe implementar el inicio de sesión con los campos de nombre de usuario y contraseña.	

4.1.3 Sprint Review

Al final del primer sprint se lleva a cabo la revisión del sprint para analizar y validar el incremento realizado, en este primer sprint se añadieron cuatro componentes nuevos a la aplicación web los cuales representan diferentes pantallas con diferentes funcionalidades, a continuación se detallan los nombres y la funcionalidad de los componentes añadidos: Autenticación, es el componente que engloba los otros tres componentes implementados y actúa como Router entre las diferentes secciones de la aplicación, Registration es el componente encargado de manejar la solicitud de envio de los campos ingresados por el usuario hacia Cognito para el registro de la cuenta, VerificationCode el Componente encargado de implementar un input para que el usuario pueda verificar su cuenta, Login es el componente que permite al usuario iniciar sesión en la aplicación. No es necesario adaptar el producto backlog. También se menciona que la funcionalidad del próximo sprint es la de Home page para la navegación del usuario autenticado.

4.1.4 Autenticación con Amplify y Código Fuente del Sprint

En lo que respecta a la iniciación del sprint se genera la configuración de amplify en la terminar que contiene la aplicación web, para ello se ejecuta el comando *amplify init* y se selecciona las opciones que mejor se adapten a las necesidades de la aplicación web, en este caso se añade el nombre del proyecto: proyecto-tit, el cual es el identificador dentro de la cuenta de AWS, se añade dev indicando el entorno de desarrollo, se utiliza Visual Studio Code como editor por defecto, se añade el lenguaje javascript ya que el lenguaje especializado para aplicaciones web, se selecciona react como framework de desarrollo, se añade src como path por defecto en donde se desarrolla la aplicación web, se agrega build como la ruta de distribución, además se agrega los comandos build y start

correspondientes a la construcción e inicialización del proyecto respectivamente.

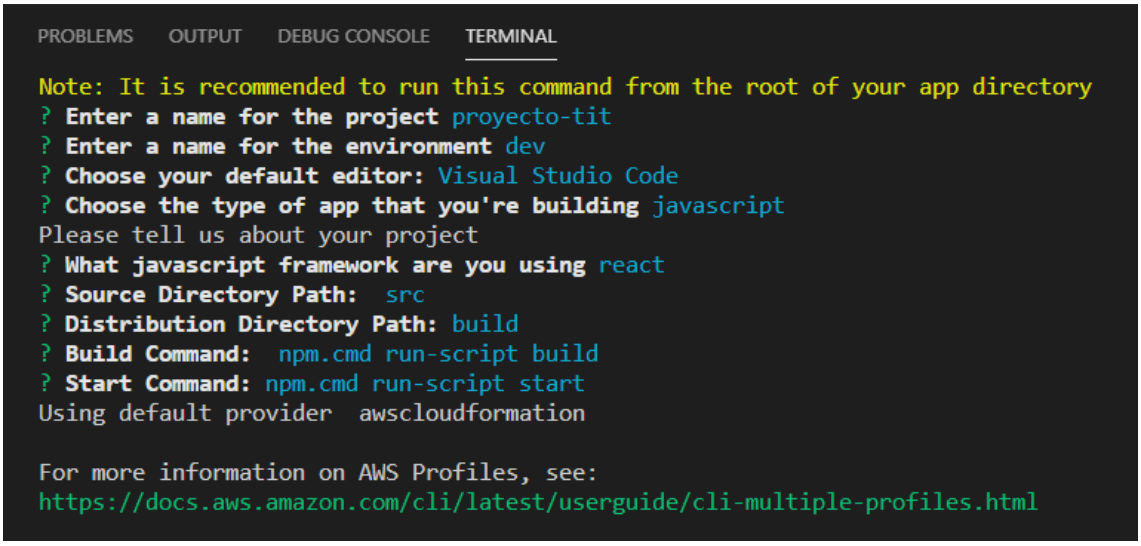


Figura 12. Integración de AWS Amplify en el Proyecto

Adicional a la configuración de Amplify se crean los recursos necesarios que serán utilizados en AWS, entre ellos se encuentran CloudFormation para modelar y aprovisionar recursos de aplicación en AWS, también se añade el servicio de S3 en donde se crea un bucket de desarrollo y finalmente se añaden los servicios de seguridad correspondientes a IAM (Identity and Access Management).

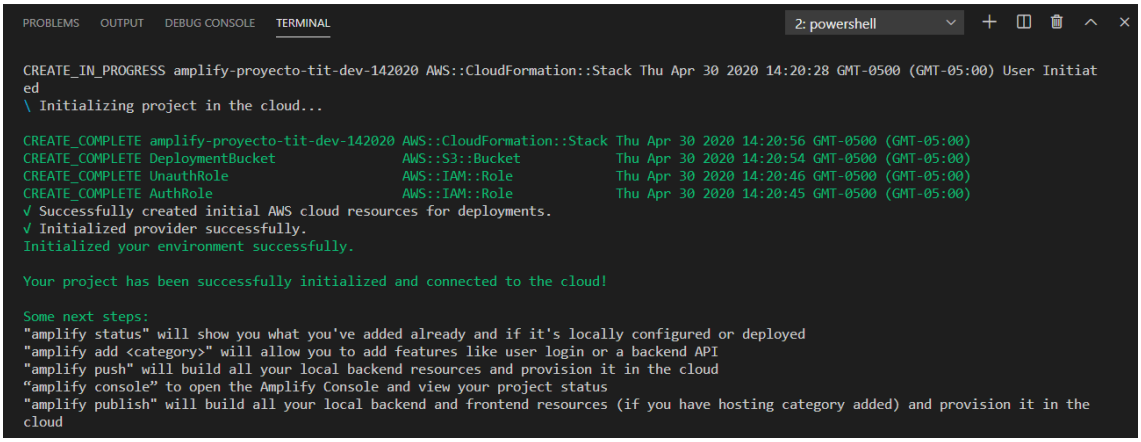


Figura 13. Creación de Recursos Cloud

Posteriormente se añade la funcionalidad de autenticación a la aplicación web mediante el comando *amplify add auth* el cual engloba el servicio AWS Cognito que pone a la disposición métodos propios para el manejo de registro, inicio de sesión, verificación y cierre de sesión.

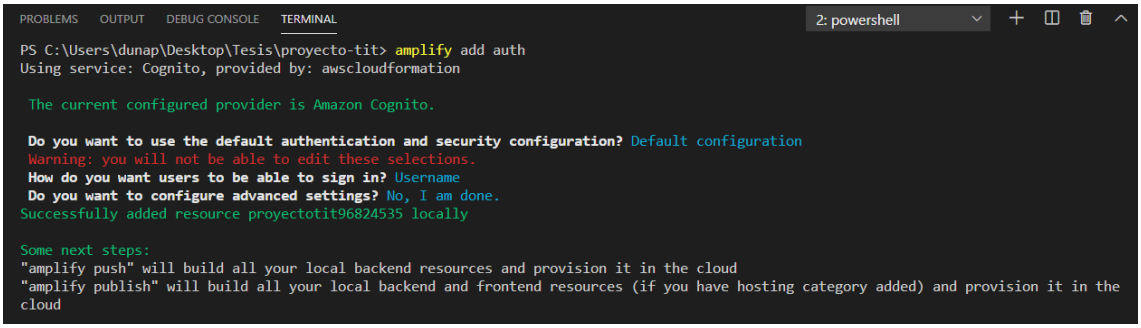


Figura 14. Implementación de Recurso Autenticación con Cognito

Posteriormente se maneja el cambio de despliegue entre los diferentes componentes relacionados a la autenticación, en este caso se hace uso de una estructura switch en donde se pasa una prop denominada switchPage la cual será la encargada de actualizar el state según el nombre que se establezca en la promesa. Se realiza cambio de componentes de esta forma debido a que el proyecto aún no posee la implementación de rutas para la carga entre diferentes componentes.

```
switch (page){
    case "SignUp":
        return(
            <RegistrationPage
                inputs={usuario}
                handleFormInput={handleFormInput}
                switchPage={handlePage}
            />
        );
    case "SignIn":
        return(
            <LoginPage
                inputs={usuario}
                handleFormInput={handleFormInput}
                switchPage={handlePage}
            />
        );
    case "Verify":
        return(
            <Verify
                inputs={usuario}
                handleFormInput={handleFormInput}
                switchPage={handlePage}
            />
        );
```

Figura 15. Renderizado de Componentes de Autenticación

Se define la función de flecha handleSignUp para implementar la funcionalidad de creación de un nuevo usuario en la aplicación web. Mediante el uso del método signUp disponible en la importación de Auth de aws-amplify es posible crear un nuevo usuario el UserPool de Amazon Cognito simplemente pasando la dirección email del nuevo usuario, la contraseña y otros atributos opcionales (Amplify Docs, s.f.).

```
const handleSignUp = e => {
  e.preventDefault();
  const { username, email, password, phone_number} = inputs;
  Auth.signUp({
    password,
    username,
    attributes:{
      email,
      phone_number
    }
  })
  .then(data => {
    console.log("Registrado");
    console.log(data);
  })
  .then(() => switchPage("Verify")) // Cambia a componente Verify
  .catch( err => console.log("Houston problemas: ", err))
}
```

Figura 16. Función de Flecha para Manejo de Promesas de Nuevos Registros

Se define la función de flecha handleVerification para implementar la funcionalidad de verificación del código al mail del usuario para comprobar la autenticidad del usuario. Para ello se hace uso del método confirmSignUp disponible en la importación de Auth de aws-amplify (Amplify Docs, s.f.).

```
const handleVerification = event => {
  event.preventDefault();
  const { username, code } = inputs;

  Auth.confirmSignUp(username, code)
  .then( data => {
      console.log("Confirmacion Exitosa");
      console.log(data);
  })
  .then(() => switchPage("SignIn"))
  .catch( err => console.log("Houston problemas: ", err))
}
```

Figura 17. Función de Flecha para Manejo de Promesas de Códigos de Verificación

Se define la función de flecha handleSignIn para implementar la funcionalidad de inicio de sesión del usuario registrado que previamente realizo la verificación de código. Mediante el uso del método signIn disponible en la importación de Auth de aws-amplify es posible autenticar al usuario (Amplify Docs, s.f.).

```
const handleSignIn = event => {
  event.preventDefault();
  const { username, password } = inputs;
  //Se puede pasar un objeto que tenga username, password y validaciones que
  //es enviado a PreAuthentication Lambda trigger
  Auth.signIn({ username, password })
    .then( user => {
      console.log("Bienvenido :)");
      console.log(user);
    })
    .then(() => switchPage("Welcome"))
    .catch( err => console.log("Houston problemas: ", err))
}
```

Figura 18. Función de Flecha para Manejo de Promesas de Inicio de Sesión

4.1.5 Entregables del Primer Sprint

En lo que respecta a los entregables del sprint se puede visualizar los componentes funcionales correspondientes a Registro, Verificación y Login respectivamente.

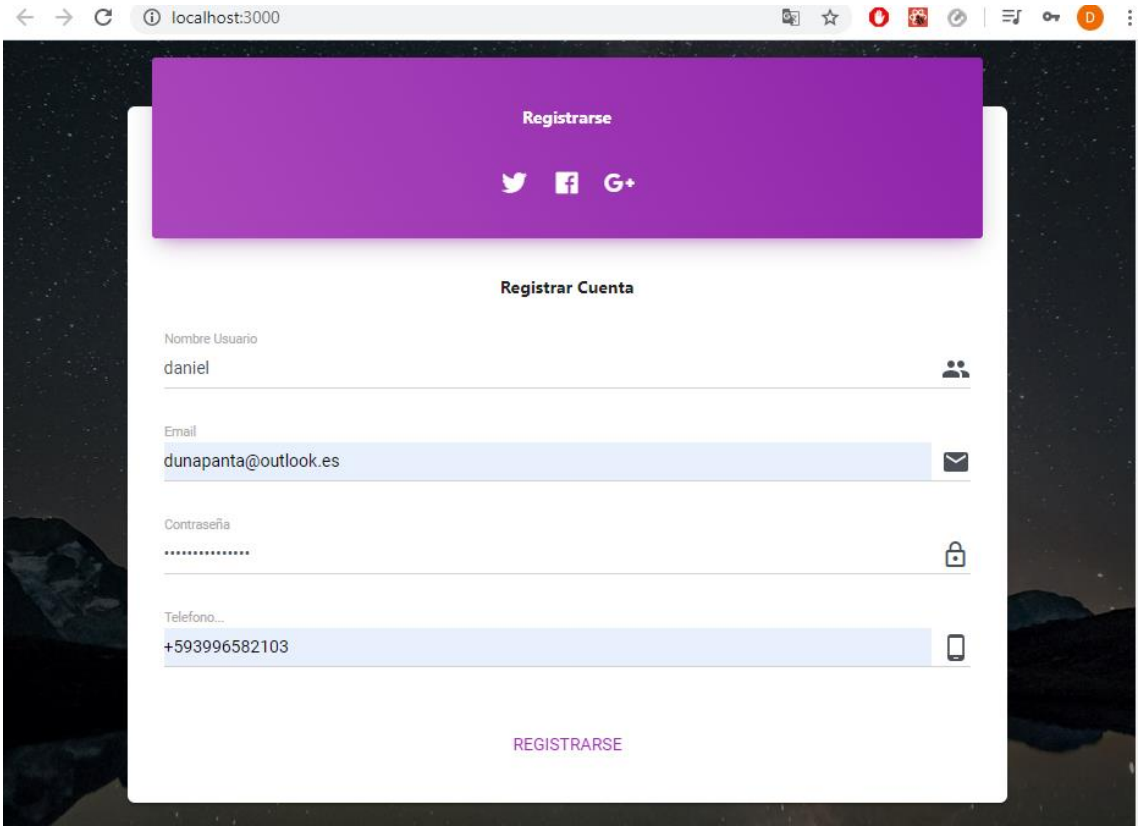


Figura 19. Renderizado de Componente de Nuevo Registro

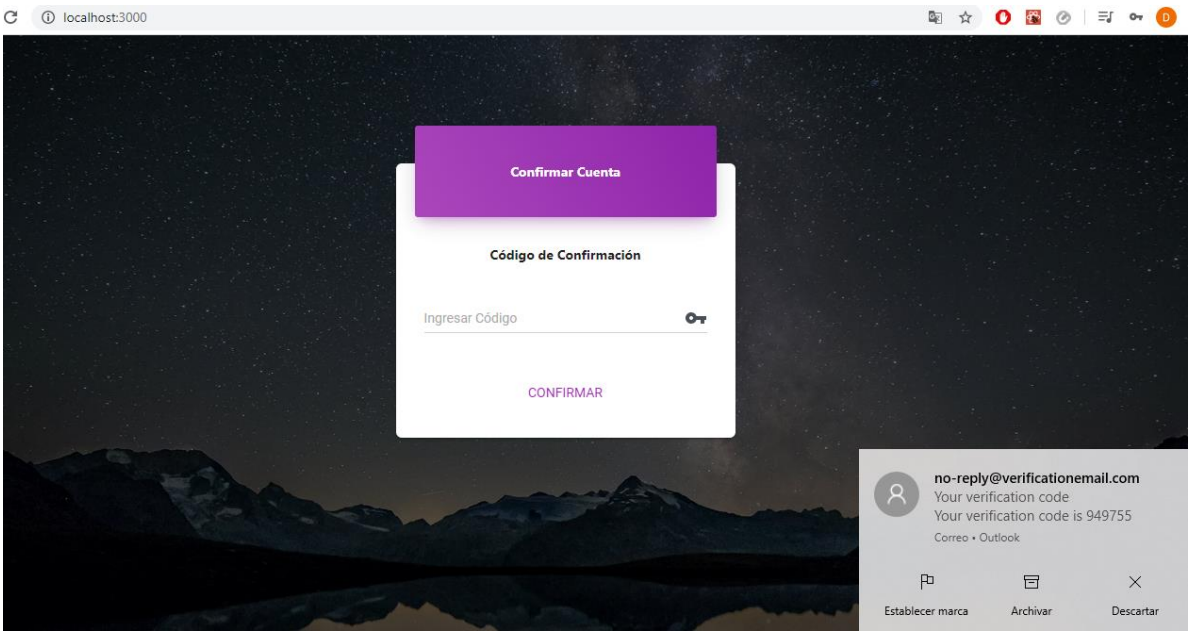


Figura 20. Renderizado de Componente de Verificación de Código

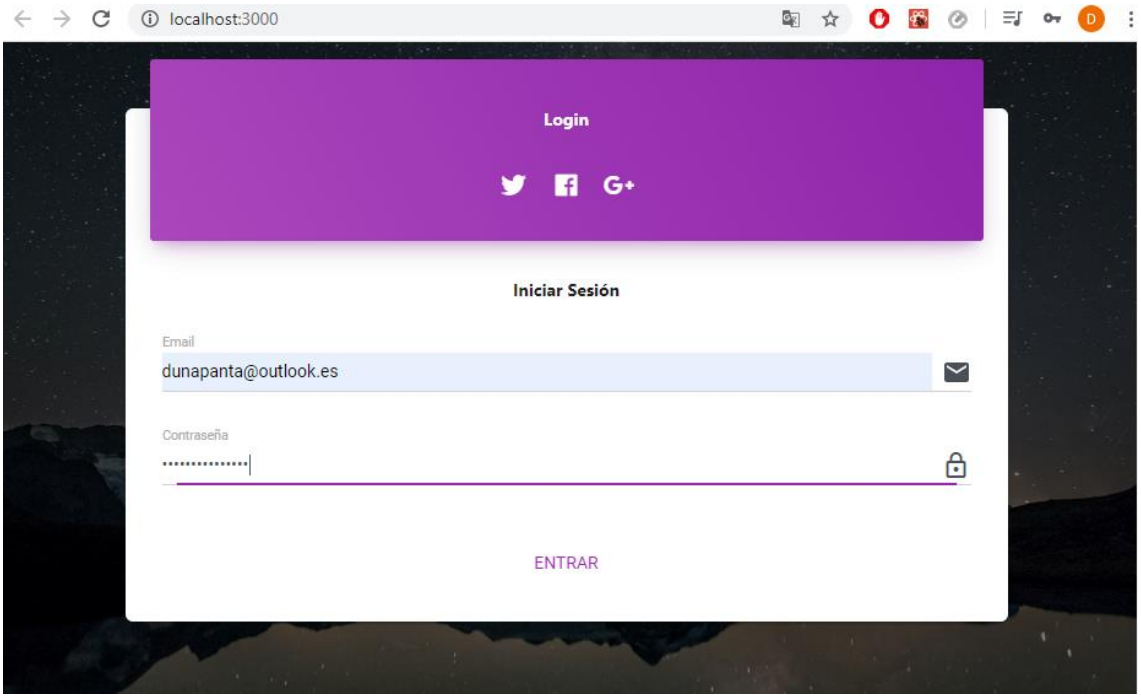


Figura 21. Renderizado de Componente de Login

Además, se visualiza el correcto registro del usuario en el userPool de Cogito en AWS.

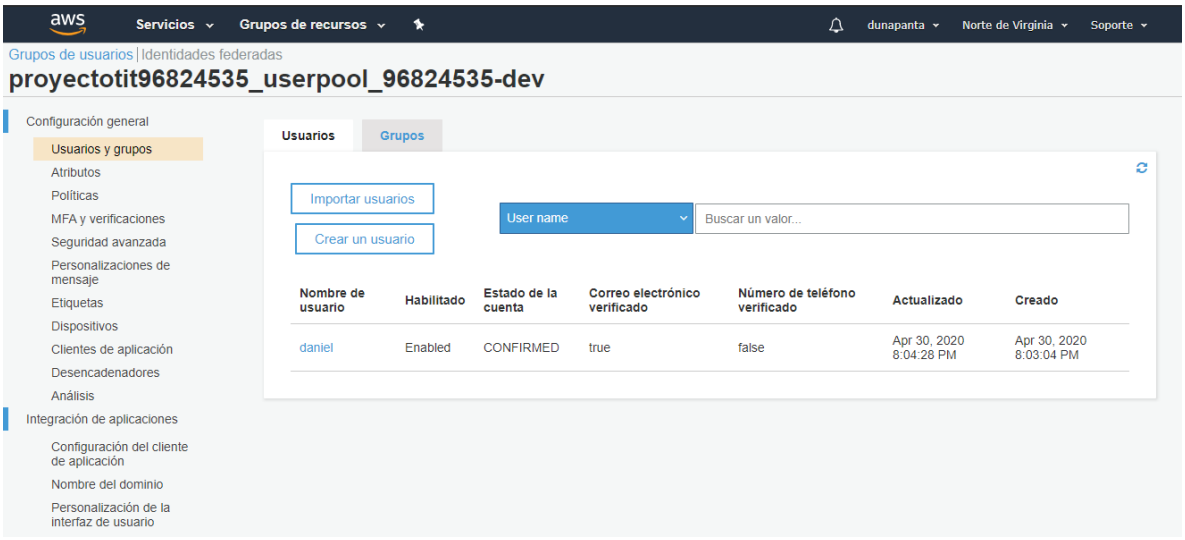


Figura 22. Verificación de la creación del usuario en Cognito

4.1.6 Sprint Restrospective del Primer Sprint

En lo que respecta al sprint restrospective del sprint realizado se establecen ciertas mejoras para entregar los entregables de los sprints más rápidamente, en este caso se decidió implementar react router para el próximo sprint para facilitar la navegación entre los diferentes componentes, además de agregar varias de las nuevas características introducidas en React 16.8, en este caso se decide implementar funcionalidades introducidas en React Hooks para los sprints ya que permiten reducir la lógica y el tiempo de codificación de los componentes.

4.2 Segundo Sprint

A continuación, se redacta el procedimiento que se llevó a cabo para la culminación del segundo sprint.

4.2.1 Sprint Planning

En lo que respecta a la reunión del sprint planning del segundo sprint, se establecen las metas y entregables que se consideran de mayor relevancia a ser implementadas en este punto del proyecto. De esta manera se establece la importancia de implementar la Home page del proyecto, la cual se accede después que el usuario inicie sesión correctamente a su cuenta desde el componente de Login. La Home page que será implementada debe constar de tres secciones principales, las cuales serán definidos en componentes separados y aislados para facilitar posibles cambios que puedan ser definidos posteriormente los componentes a considerar son los siguientes:

- **HeaderComponent:** El Componente HeaderComponent debe estar implementado al inicio de Home page, en este debe estar definido el navbar que contendrá los botones que direccionarán a diferentes funcionalidades de la aplicación web, además en HeaderComponent también se define el título y la presentación de la aplicación web que conforma la interfaz de inicio a la que el usuario accede por primera vez.
- **ContentComponent:** En el componente ContentComponent se presenta la información de las características y funcionalidades que están accesibles en la aplicación web con el fin de dar a conocer y facilitar al usuario el uso de la aplicación web.
- **FooterComponent** En el FooterComponent se define el componente correspondiente al pie de página con el fin de exponer información de contacto y datos relevantes de la aplicación web.

4.2.2 Historias de Usuario del Sprint Backlog del Segundo Sprint

Tabla 11.

Historia de Usuario Barra de Navegación

Historia de usuario	
ID: HOME_01	**Usuario:** Daniel Unapanta
Nombre de historia: Barra de Navegación	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 2PH	**Sprint asignado:** 2
Responsable: Daniel Unapanta	
Descripción: Como empleador y trabajador, deseo tener links de acceso en un navbar, para acceder a funcionalidades de la aplicación	
Criterios de aceptación: Se debe proporcionar una barra de navegación para facilidad de uso y acceso a los servicios de definición de curriculum, acceso a categorías, despliegue de contrataciones y cierre de sesión.	

Tabla 12.

Historia de Usuario Contenido de HomePage

Historia de usuario	
ID: HOME_02	**Usuario:** Daniel Unapanta
Nombre de historia: Contenido de Homepage	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 2PH	**Sprint asignado:** 2
Responsable: Daniel Unapanta	
Descripción: Como empleador y trabajador, deseo conocer las funcionalidades y características que provee la aplicación web para facilitar el uso de la misma.	
Criterios de aceptación: Se debe definir el contenido principal referente a los fines de la aplicación web.	

Tabla 13.

Historia de Usuario Pie de Página

Historia de usuario	
ID: HOME_03	**Usuario:** Daniel Unapanta
Nombre de historia: Pie de Página	
Prioridad en negocio: Media	**Riesgo en desarrollo:** N/A
Estimación: 1PH	**Sprint asignado:** 2
Responsable: Daniel Unapanta	
Descripción: Como empleador y trabajador, deseo tener disponible información de contacto y de copyrigth de la aplicación web para conocer datos relevantes de la aplicación web.	
Criterios de aceptación: Se debe definir el pie de página.	

4.2.3 Sprint Review

Se lleva a cabo la revisión del segundo sprint para analizar y validar el incremento realizado, se añadieron cuatro nuevos componentes a la aplicación web los cuales se detallan a continuación:

- **HeaderComponent:** Este componente fue definido con funcionalidades de React Router en donde se utilizan varios componentes de Material UI como son: AppBar, ToolBar, Buton, GridContainer, GridItem, List y ListItem. En este componente se definen cuatro botones: Categorías, Buscar Trabajo, Mi Curruculum y Github, los primeros tres representan funcionalidades que serán implementadas posteriormente y el ultimo representa un link para el perfil de github que contiene la aplicación web, también se hace uso de un componente personalizado denominado Parallax para presentr la interfaz de Header al usuario.
- **ContentComponent:** El componente ContentComponent se implementa mediante el uso de varios componentes de Material UI como son: InfoArea, GridContainer y GridItem además de los

iconos svg WorkIcon, EmojiPeopleIcon y VerifiedUser, de igual manera el componente se lo define utilizando React Hooks. La información proporcionada se la divide mediante GridItem para facilitar el despliegue de información tanto en dispositivos de escritorio, tablets o móviles y se define las características de Trabajador, Empleador y Plataforma.

- **FooterComponent:** Este componente se define mediante el uso de List y ListItem de Material UI en él se presentan información de la aplicación y derechos de copyrigth.
- **HomeComponent:** Este es el componente en donde se importan los componentes previamente definidos en este sprint, es decir es el componente padre el cual se encarga de pasar las props según las necesidades definidas de cada componente hijo. Este es el componente que de despliega cuando se navega a la ruta /home previamente definida con react-router.

4.2.4 Código Fuente del Segundo Sprint

El Sprint comienza con la implementación de Rutas que serán definidas y podrán ser accedidas a través del navbar dentro de HeaderComponent o a través de programación por medio de promesas.

```
return(
    <Switch>
        <Route path="/login">
            <LoginPage
                inputs={usuario}
                handleFormInput={handleFormInput}  />
        </Route>
        <Route path="/register">
            <RegistrationPage
                inputs={usuario}
                handleFormInput={handleFormInput}  />
        </Route>
        <Route path="/verify">
            <Verify
                inputs={usuario}
                handleFormInput={handleFormInput} />
        </Route>
        <Route path="/home">
            <h1>Pin Pan Pun Logeado</h1>
        </Route>
        <Route path="/">
            <RegistrationPage
                inputs={usuario}
                handleFormInput={handleFormInput}  />
        </Route>
    </Switch>
```

Figura 23. Definición de Rutas con react-router

Definición de HeaderComponent en donde se implementan varios botones de presentación correspondientes al Navbar

```
return (
  <List className={classes.list}>
    <ListItem className={classes.listItem}>
     <Link to="/register" style={{ color: 'inherit', textDecoration: 'inherit' }}>
      <Button
        color="transparent"
        target="_blank"
        className={classes.navLink}
      >
        <PersonAdd className={classes.icons} /> REGISTRARSE
      </Button>
      </Link>
    </ListItem>
```

Figura 24. HeaderComponent definiendo Navbar

Implementación de ContentComponente en donde se establecen varios grid para la presentación de características de la aplicación.

```
<GridItem xs={12} sm={12} md={4}>
  <InfoArea
    title="Trabajador"
    description="El usuario de la aplicación es capaz de definir su
    icon={WorkIcon}
    iconColor="info"
    vertical
  />
</GridItem>
<GridItem xs={12} sm={12} md={4}>
  <InfoArea
    title="Empleador"
    description="De igual manera el usuario es capaz de buscar entre
    icon={EmojiPeopleIcon}
    iconColor="danger"
    vertical
  />
</GridItem>
```

Figura 25. ContentComponent definición de Grids

Definición del return del componente denominado FooterComponent en donde se hace uso de varios estilos previamente definidos para presentación de información y copyrigth de la aplicación web.

```
return (
  <footer className={footerClasses}>
    <div className={classes.container}>
      <div className={classes.left}>
        <List className={classes.list}>
          <ListItem className={classes.inlineBlock}>
            <a
              href="https://github.com/dunapanta/Proyecto-TIT"
              className={classes.block}
              target="_blank"
            >
              Proyecto Titulación
            </a>
          </ListItem>
        </List>
      </div>
      <div className={classes.right}>
        &copy; {1900 + new Date().getYear()} , Hecho por Daniel Unapanta
      </div>
    </div>
  </footer>
);
```

Figura 26. FooterComponent información y copyrigth

4.2.5 Entregables del Segundo Sprint

A continuación, se presenta el renderizado de los componentes HeaderComponent, ContentComponent y FooterComponent respectivamente.

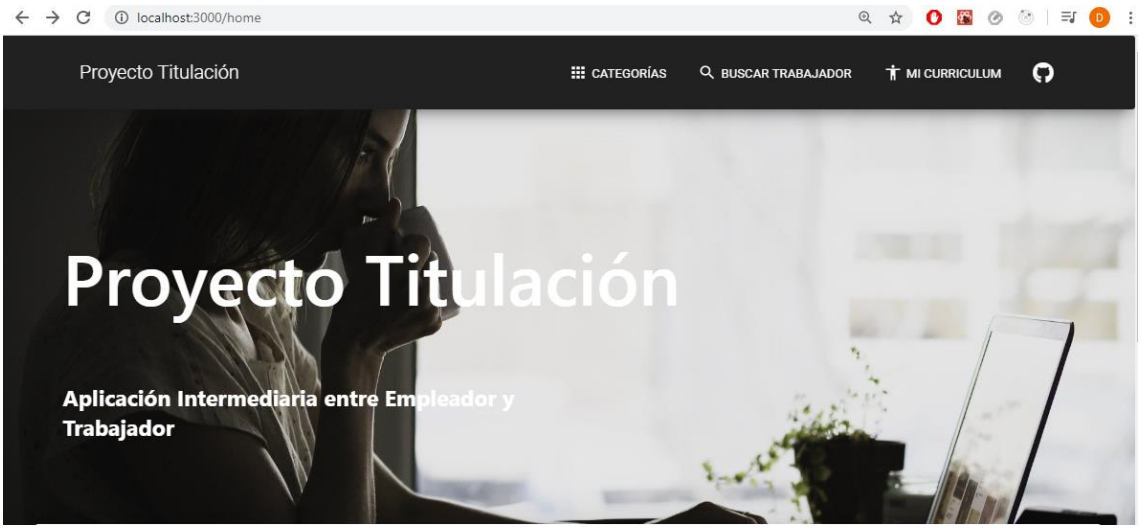


Figura 27. Renderizado de HeaderComponent

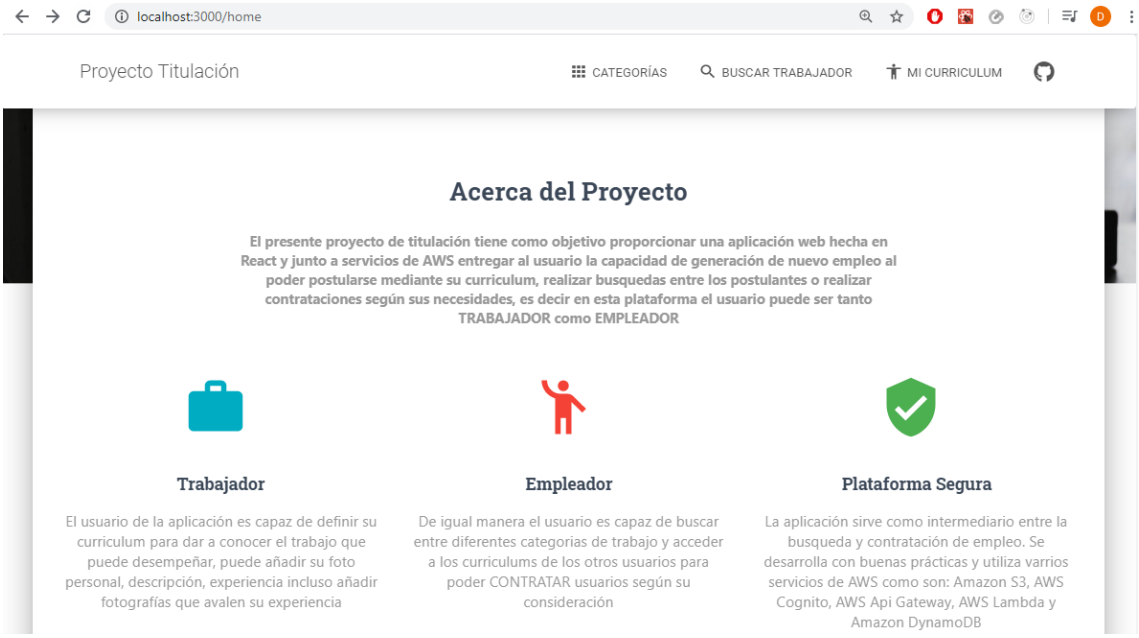


Figura 28. Renderizado de ContentComponent

Figura 29. Renderizado del FooterComponent

4.2.6 Sprint Restrospective del Segundo Sprint

En lo que respecta al sprint restrospective del segundo sprint realizado se establecen ciertas mejoras para entregar los entregables de los sprints más rápidamente, en este caso se decidió implementar react router para el próximo sprint para facilitar la navegación entre los diferentes componentes, además de agregar varias de las nuevas características introducidas en React 16.8, en este caso se decide implementar funcionalidades introducidas en React Hooks para los sprints ya que permiten reducir la lógica y el tiempo de codificación de los componentes.

4.3 Tercer Sprint

A continuación, se redacta el procedimiento que se llevó a cabo para la culminación del tercer sprint.

4.3.1 Sprint Planning

En lo que respecta a la reunión del sprint planning del tercer sprint, se instauran las metas y entregables que son necesarios a implementar en este punto del proyecto. Es así que se considera relevante la implementación del componente Curriculum correspondiente a la creación o actualización del curriculum del usuario registrado, el componente debe ser accedido por medio de la implementación de la ruta curriculum, previamente definida. El componente curriculum se define a partir de la reutilización de los componentes Header y Footer creados en el segundo sprint y también se crean dos componentes nuevos definidos en archivos independientes en los cuales establecerán comunicación por medio de props, los componentes son: Formulario Curriculum correspondiente a los

datos considerados relevantes del usuario y Fotografía Curriculum para la actualización de la fotografía del usuario.

4.3.2 Historias de Usuario del Sprint Backlog del Tercer Sprint

Tabla 14.

Historia de Usuario Formulario Curriculum

Historia de usuario	
ID: CUR_01	**Usuario:** Daniel Unapanta
Nombre de historia: Formulario Curriculum	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 8PH	**Sprint asignado:** 3
Responsable: Daniel Unapanta	
Descripción: Como trabajador, deseo mantener mi curriculum personal actualizado, para dar a conocer mis habilidades, destrezas y experiencias a un posible empleador.	
Criterios de aceptación: Se debe proporcionar la sección de definición de trabajo en donde el usuario es capaz de ingresar los datos de su curriculum, entre los campos a ingresar constan: nombres, apellidos, cédula, teléfono, categoría de trabajo, nombre trabajo, tarifa, ciudad, país, código postal, acerca de mí y experiencia laboral.	

Tabla 15.

Historia de Usuario Fotografía Curriculum

Historia de usuario	
ID: CUR_02	**Usuario:** Daniel Unapanta
Nombre de historia: Fotografía Curriculum	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 5PH	**Sprint asignado:** 3
Responsable: Daniel Unapanta	

Descripción: Como trabajador, deseo actualizar mi fotografía de perfil, para facilitar la elección y contratación de un posible empleador.
Criterios de aceptación: Se debe proporcionar la sección de fotografía de curriculum en donde el usuario puede subir una fotografía personal en formato jpg o png.

4.3.3 Sprint Review

Se lleva a cabo la revisión del tercer sprint para analizar y validar el incremento realizado, se añadieron dos nuevos componentes a la aplicación web los cuales se detallan a continuación:

- **FormularioCurriculum:** Este componente fue definido con la utilización de diferentes componentes de Material UI, entre los principales se encuentran GridContainer, GridItem, Card, CardHeader, BardBody, CardFootter y CustomInputs. En este componente se definen el manejo del formulario que es llenado por el usuario, en él se definen varios campos relevantes de información de usuario: Nombres, Apellidos, Cédula, Télefono, Categoría Trabajo, Nombre Trabajo, Tarifa en $, Ciudad, País, Código Postal, Descripción Personal y Experiencia laboral. Cada uno de los campos se maneja mediante el uso del hook useState el cual permite almacenar el estado de los datos para ser enviados a Dynamodb.
- **FotografiaCurriculum:** El componente FotografiaCurriculum se implementa mediante el uso de varios componentes de Material UI como son: GridItem, GridContainer, Card, CardHeader, CardBody, CardFooter, Button y CustomInput. En este componente se hace uso del servicio proporcionado por Amplify para obtener, almacenar y enviar archivos y documentos a Amazon S3 para ello se establecen promesas, se hace uso de los hooks useState y useEffect para definir la lógica del componente.

4.3.4 Código Fuente del Tercer Sprint

Definición de la promesa correspondiente a la comunicación entre los datos almacenados en useState y el envió de datos por medio de la API previamente definida, el medio por el cual se almacenará la información del usuario en Dynamodb.

```
const submitCurriculum = () => {
    const {usuario} = props;
    console.log(userCurriculum);
    console.log("El usuario es:", usuario);
    console.log("El id es:", usuario.user.sub);
    console.log("atributos:", userCurriculum);
    let apiName = "pruebatesis";
    let path = "/user";
    let data = {
        body: {
            user_id: usuario.user.sub,
            ...userCurriculum
        }
    };
    console.log("La data ecrita:", data);
    API.post(apiName, path, data)
        .then(response => {
            console.log(response)
        })
        .catch(error => {
            console.log(error.response)
        })
```

Figura 30. Promesa comunicación con API

Definición de método con actualización del estado por medio de useState para desplegar la fotografía especifica del usuario

```
const getProfilePictureAsync =  async () => {
  console.log("Desde Await", username)
  const storage =  await Storage.get(`${username.username.username}.png`);
  console.log("Desde Await", username)
  console.log("Storage",storage)
  try{
  setImage(storage);
  }catch(err){
    console.log("Error peticion foto", err)
    setImage(avatar)
  }
}
```

Figura 31. Método de obtención de imagen de usuario.

Definición del método que define la promesa de comportamiento asíncrono para enviar la imagen previamente seleccionada para ser enviada al bucket de Amazon S3.

```
const onProcessFile = e => {
  e.preventDefault();
  let reader = new FileReader();
  let file = e.target.files[0];
  try {
    reader.readAsDataURL(file);
  } catch (err) {
    console.log("Error Fotografia", err);
  }
  reader.onloadend = () => {
    setImage(reader.result);
  };
  Storage.put(`${username.username}.png`, file, {
    contentType: "image/png"
  })
    .then(result => console.log(result))
    .catch(err => console.log(err));
};
```

Figura 32.Promesa de envió a bucket de Amazon S3

4.3.5 Entregables del Tercer Sprint

A continuación, se presenta una breve descripción de los entregables del tercer sprint:

En primer lugar, se establece la actualización de HeaderComponent en donde se define opciones de sesión al usuario para facilitar el uso de la plataforma.

Figura 33. Actualización HeaderComponente en Curriculum

Se define el componente FotografiaComponent en donde al presionar el botón seleccionar fotografía permite al usuario elegir su fotografía personal en el sistema de archivos de su dispositivo y para enviarla y almacenarla en Amazon S3 se presiona el botón Subir Fotografía.

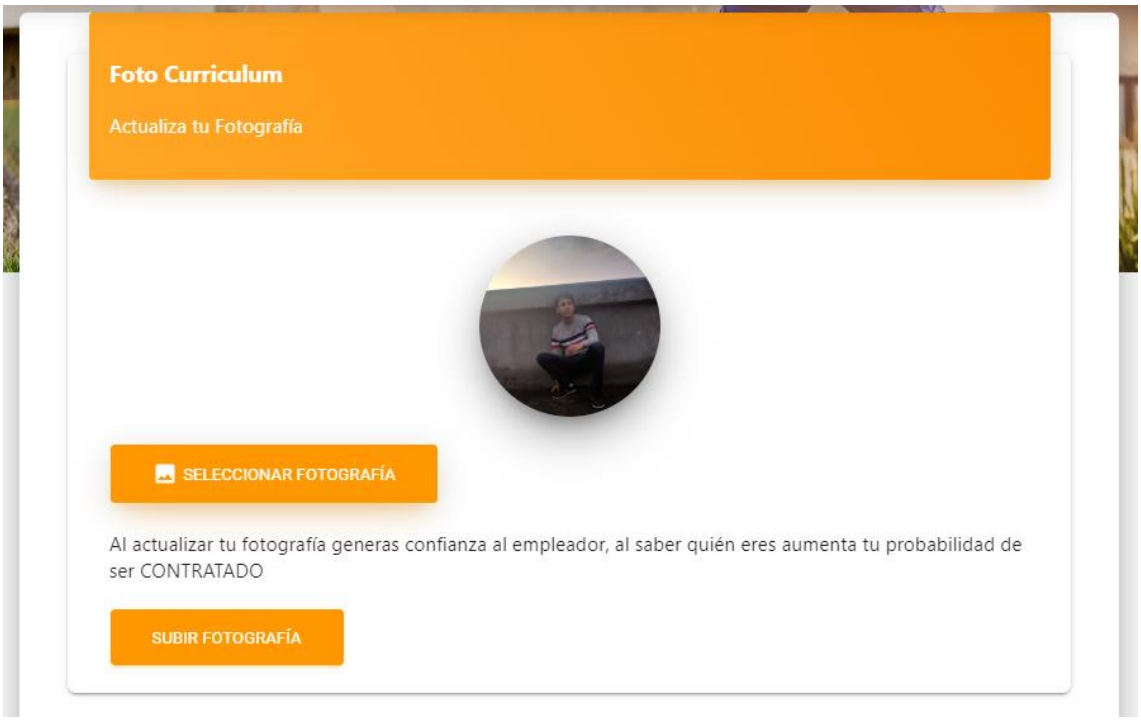


Figura 34. Despliegue de FotografiaComponent

Se verifica el almacenamiento de la fotografía del usuario ingresando al bucket correspondiente en Amazon S3

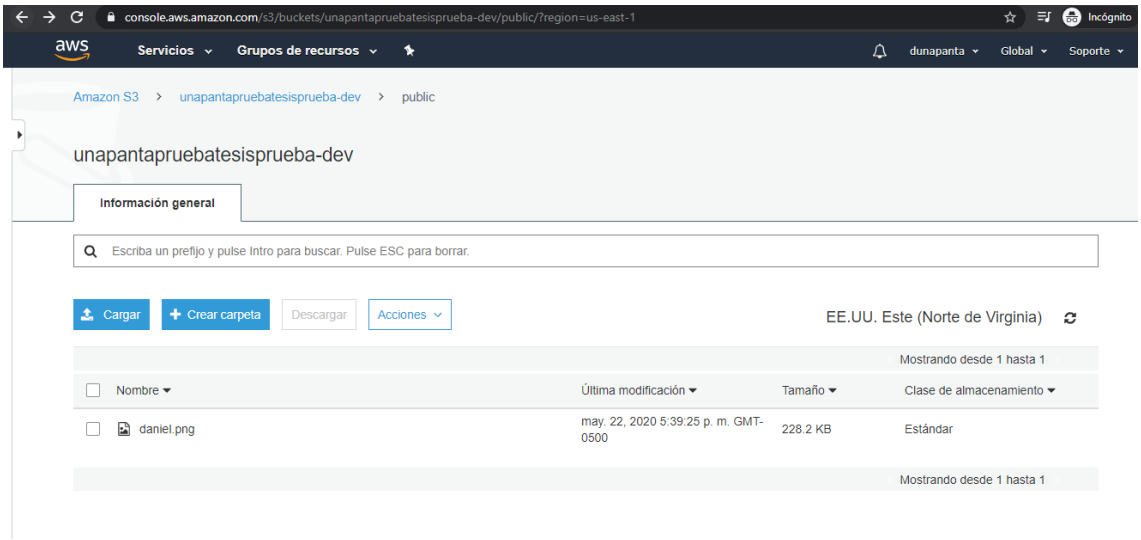


Figura 35. Verificación de Almacenamiento en Amazon S3

Se define el componente CurriculumComponent en el cual se establece una serie de campos que el usuario debe ingresar para completar su curriculum personal.

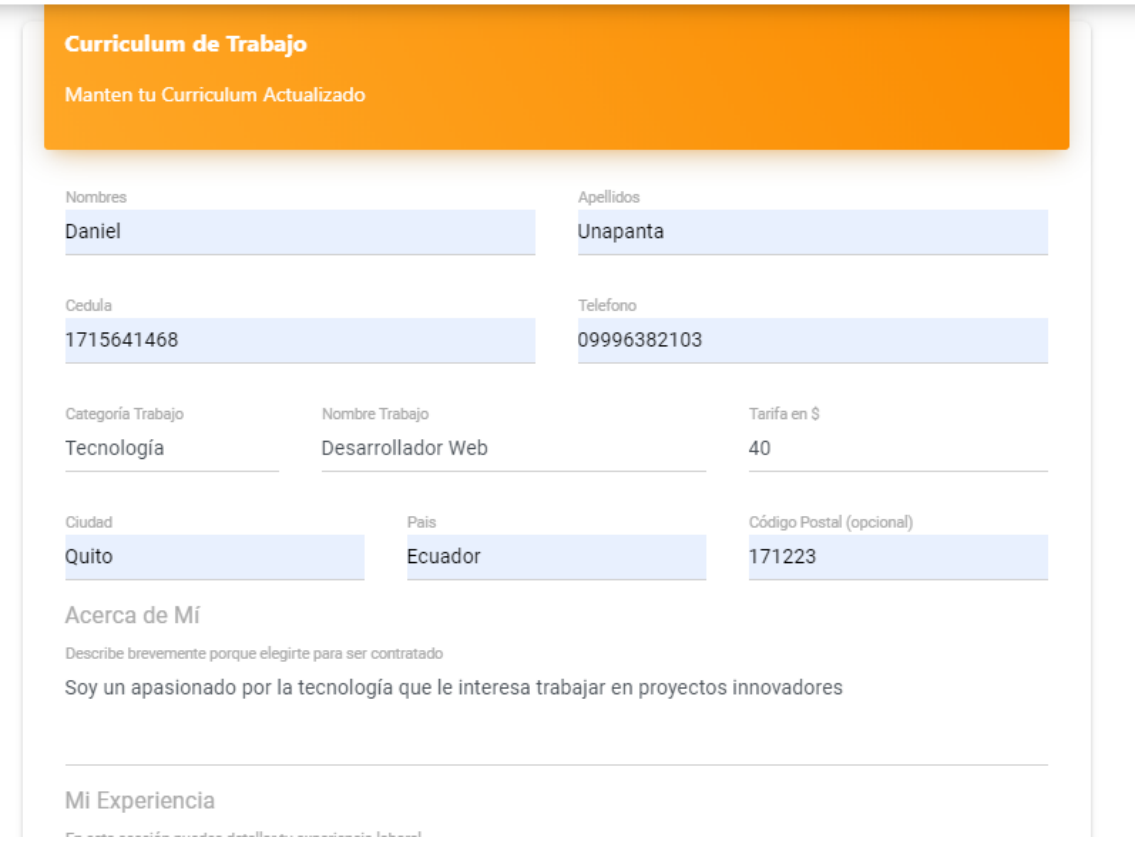


Figura 36. Despliegue de FormularioComponent

Se verifica el correcto almacenamiento de los datos proporcionados por el usuario y que estén almacenados en el servicio DynamoDb de Amazon.

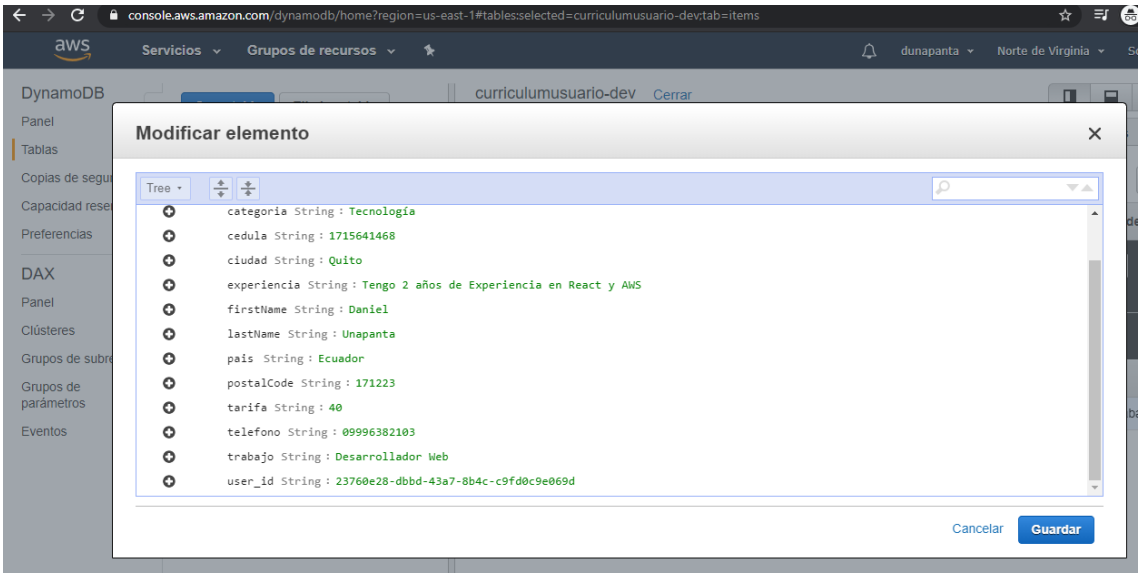


Figura 37. Verificación de Almacenamiento de Curriculum en DynamoDB

4.3.6 Sprint Restrospective del Tercer Sprint

En lo que respecta al sprint restrospective del tercer sprint se establecen ciertas mejoras a implementar en el proyecto, en este caso el modularidad de ciertos componentes que definen gran parte de la lógica en la comunicación de la API definida en API Gateway, es necesario dividir la lógica en varios métodos que podrán ser usuarios de manera más sencilla, también se analiza la posibilidad de utilización de hooks como useMemo y useContext para mejorar el rendimiento de la aplicación.

4.4 Cuarto Sprint

A continuación, se redacta el procedimiento que se llevó a cabo para la culminación del segundo sprint.

4.4.1 Sprint Planning

En lo que respecta a la reunión del sprint planning del cuarto sprint, de manera similar tal y como se ha venido trabajando se imparten las metas y entregables a implementar en esta etapa del proyecto. Los componentes por desarrollar constituyen la funcionalidad de categorías en donde el usuario debe ser capaz de acceder a la Categoría que conforman un conjunto de trabajos definidos en la lógica de la aplicación y de esta manera poder acceder al listado de usuarios disponibles para ese trabajo, es necesario implementar el CategoriesComponent, el cuál debe ser accedido por medio de un dropdown menú y por medio de rutas definidas. También debe definirse accesos directos para filtrar los trabajadores de una categoría en específico. El componente CategoriesComponent se constituye a partir de la reutilización de los componentes Header y Footer y también es necesario introducir un nuevo componente para el mapeo de todos los usuarios registrados que cumplen con la categoría de trabajo seleccionada.

4.4.2 Historias de Usuario del Sprint Backlog del Cuarto Sprint

Tabla 16.

Historia de Usuario DropDown Menu Categorías

Historia de usuario	
ID: CAT_01	**Usuario:** Daniel Unapanta
Nombre de historia: DropDown Menu Categorías	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 2PH	**Sprint asignado:** 4
Responsable: Daniel Unapanta	

Descripción: Como empleador, deseo desplegar un listado con las categorías más relevantes, para acceder fácilmente al listado de los trabajadores registrados correspondientes a la categoría seleccionada
Criterios de aceptación: Se debe proporcionar un menú desplegable con el nombre de las categorías de trabajos disponibles en donde se filtran los trabajadores por categoría.

Tabla 17.

Historia de Usuario Despliegue de Perfiles de Trabajadores

Historia de usuario	
ID: CAT_02	**Usuario:** Daniel Unapanta
Nombre de historia: Despliegue de Perfiles de Trabajadores	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 3PH	**Sprint asignado:** 4
Responsable: Daniel Unapanta	
Descripción: Como trabajador, deseo desplegar el listado de los trabajadores correspondientes a una categoría seleccionada, para verificar los posibles candidatos a ser contratados.	
Criterios de aceptación: Se debe proporcionar la sección despliegue de trabajadores, para cada trabajador se debe incluir la información de: trabajo, categoría, fotografía personal, nombre, tarifa, acerca de mí y un botón de visualización de perfil.	

4.4.3 Sprint Review

Se lleva a cabo la revisión del cuarto sprint para analizar y validar el incremento realizado, se añadieron cuatro nuevos componentes a la aplicación web los cuales se detallan a continuación:

- **CategoriesComponent:** Este es el componente principal del presente Sprint, en el cual se despliegan todos los perfiles de los trabajadores registrados en la aplicación web. Este

componente se define a partir de la reutilización de los componentes HeaderComponent y FooterComponent además se utilizan los componentes de MaterialUI: GridContainer, GridItem, Card, CardHeader, CardBody, CardAvatar. En este componente se evidencia la necesidad de redefinir la función Lambda correspondiente a la obtención de la lista de objetos de los perfiles registrados con curriculum. Por medio de useState se implementan estados para la lista de usuarios, fotografías y loading, de igual forma se hace uso del hook useEffect para la obtención los datos requeridos por medio de Amazon API Gateway.

- **CategoryEducation:** Este componente es el encargado de filtrar los resultados obtenidos para la categoría Educación, es decir que en este componente se despliegan todos los usuarios que definieron su curriculum con la categoría Educación, la filtración de los resultados se lo realiza por medio de la función map definida en javascript para los arreglos.
- **CategoryHome:** Este componente es el encargado de filtrar los resultados obtenidos para la categoría Hogar, de manera similar solo despliega los usuarios con la categoría Hogar en su curriculum, la filtración de los resultados se lo realiza por medio de la función map definida en javascript para los arreglos.
- **CategoryTecnology** Este componente es el encargado de filtrar los resultados obtenidos para la categoría Tecnología, de igual manera solo aparecen los usuarios que definieron la categoría Tecnología en su curriculum, la filtración de los resultados se lo realiza por medio de la función map definida en javascript para los arreglos.

4.4.4 Código Fuente del Cuarto Sprint

Definición de función Lambda para la obtención de la lista de todos los usuarios que definieron su curriculum, para esta sección de código se

implementó el método scan disponible en dynamodb para la obtención de una lista con todos los datos requeridos (Klems, 2018).

```
app.get(path, function(req, res) {
  var condition = {}
  condition[partitionKeyName] = {
    ComparisonOperator: 'EQ'
  }
  if (userIdPresent && req.apiGateway) {
    condition[partitionKeyName]['AttributeValueList'] = [req.apiGateway.event.requestContext.identity.
    cognitoIdentityId || UNAUTH ];
  } else {
    try {
      condition[partitionKeyName]['AttributeValueList'] = [ convertUrlType(req.params[partitionKeyName],
      partitionKeyType) ];
    } catch(err) {
      res.statusCode = 500;
      res.json({error: 'Wrong column type ' + err});
    }
  }
  let queryParams = {
    TableName: tableName,
    KeyConditions: condition
  }
  dynamodb.scan(queryParams, (err, data) => {
    if (err) {
      res.statusCode = 500;
      res.json({error: 'Could not load items: ' + err});
    } else {
```

Figura 38. Función Lambda para Obtención Lista de Usuarios con Curriculum.

Definición de función para la petición y asignación de la lista de usuarios al estado del componente por medio del hook useState previamente definido.

```
const getUserCurriculumAsync = async () => {

    let path = "/user";
    const apiName = "pruebatesis";
    let myInit = {
        headers: {},
        queryStringParameters: {
        },
        response: true,
    }
    const response  = await API.get(apiName, path, myInit);

    setListWorkers(response);

    console.log("listWorkers", listWorkers);
    console.log("0 de listWorkers", listWorkers[0]);

    setLoading(false);
}
```

Figura 39. Obtención y Asignación del Listado al Estado del Componente

Definición del filtrado de usuarios definidos por determinada categoría en curriculum, se hace uso de la función map de Javascript para iterar en cada elemento del arreglo de usuarios.

```
{listWorkers.map(trabajador => {
if(trabajador.categoria === "Educacion"){
    return (
    <GridItem key={trabajador.user_id} xs={12} sm={12} md={4}>
        <Card profile>
            <CardHeader color="primary">
    <h3 className={classes.cardTitleWhite}>Trabajo: {trabajador.trabajo}</h3>
    <p className={classes.cardCategoryWhite}>Categoría: {trabajador.categoria}</p>
            </CardHeader>
            <CardAvatar profile>
            <a  href="#pablo" onClick={e => e.preventDefault()}>
                <img src={renderPhoto(trabajador.user.username)} alt="..." />
            </a>
            </CardAvatar>
            <CardBody profile>
    <strong style={ {display: "block", fontSize: "18px", textAlign: "center", paddi
    <strong style={ {display: "block", fontSize: "14px", textAlign: "center", margi
                <p className={classes.description}>
                    {trabajador.aboutMe}
                </p>
                <Button color="primary">
                    Ver Perfil del Usuario
                </Button>
                </CardBody>
        </Card>
```

Figura 40. Definición de Función para Filtrar por Categoría

4.4.5 Entregables del Cuarto Sprint

A continuación, se presenta una breve descripción de los entregables del cuarto sprint:

Como primer entregable se define la actualización del HeaderComponent que será utilizado en CategoriesComponent.

Figura 41. Actualización HeaderComponente en Categorías

Despliegue de Categorías disponibles en la aplicación por medio de un DropDown Menu

Figura 42. DropDown Menú de Categorías

Despliegue de todos los usuarios que definieron su curriculum, en este componente se visualiza la lista renderizada de cada uno de los usuarios registrados con curriculum definido.

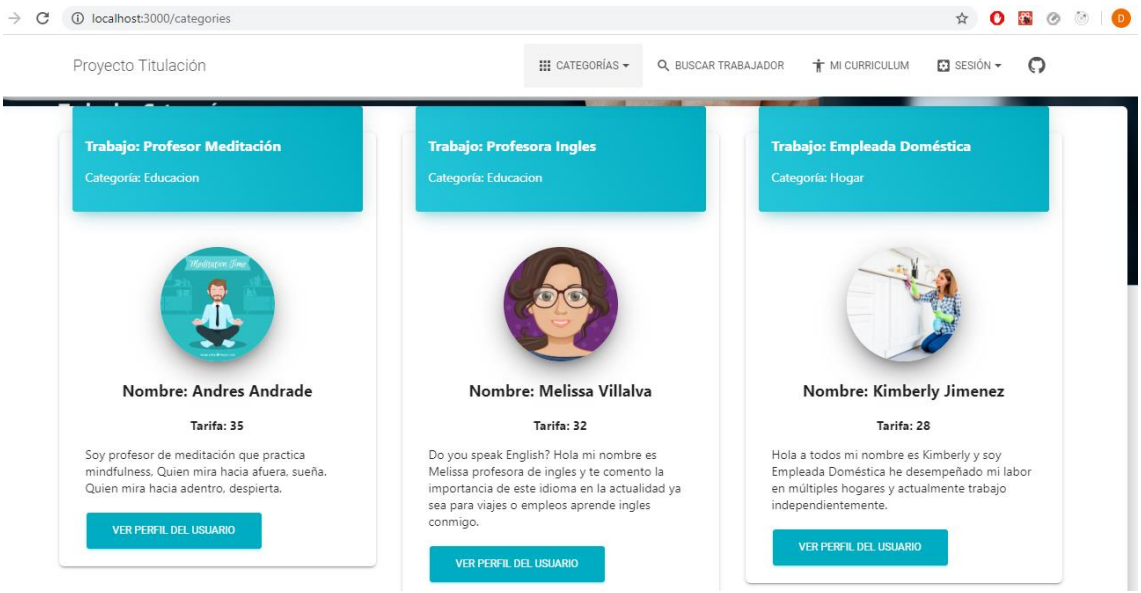


Figura 43. Despliegue de Usuarios de Todas las Categorías

Despliegue de usuarios filtrados con categoría Educación

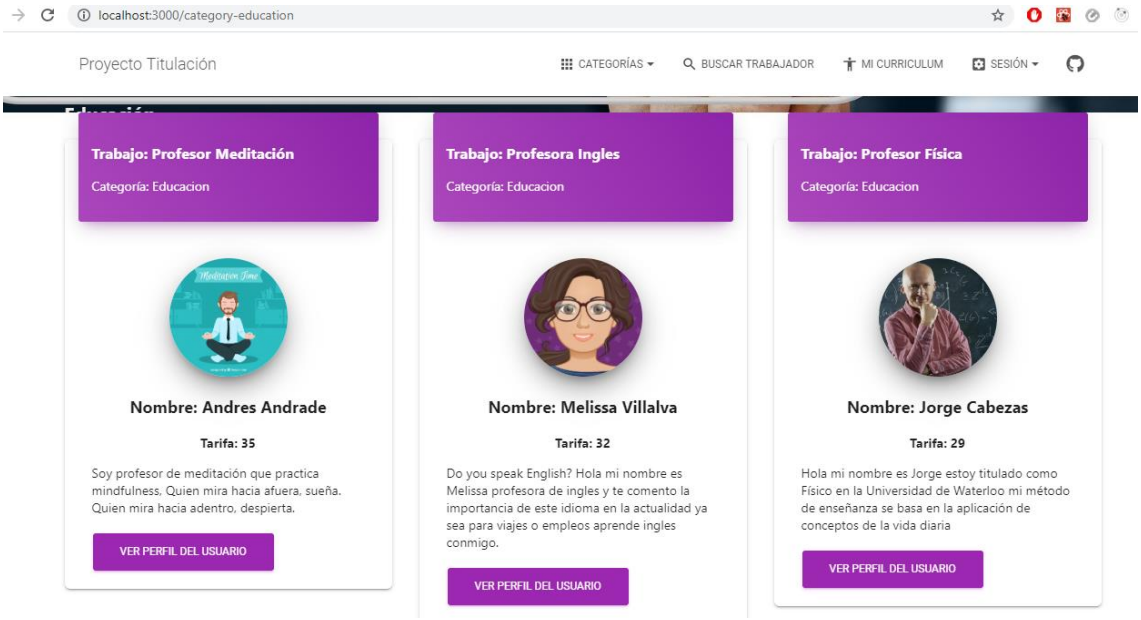


Figura 44. Despliegue de Usuarios con Categoría Educación

Despliegue de usuarios filtrados con categoría Hogar

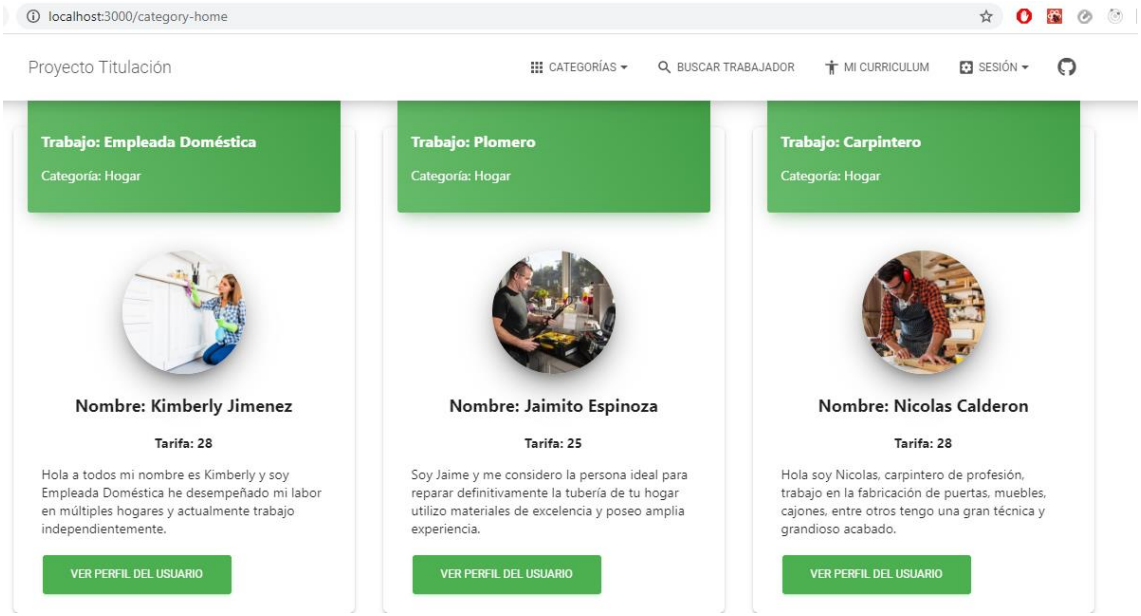


Figura 45. Despliegue de Usuarios con Categoría Hogar

Despliegue de usuarios filtrados con categoría Tecnología

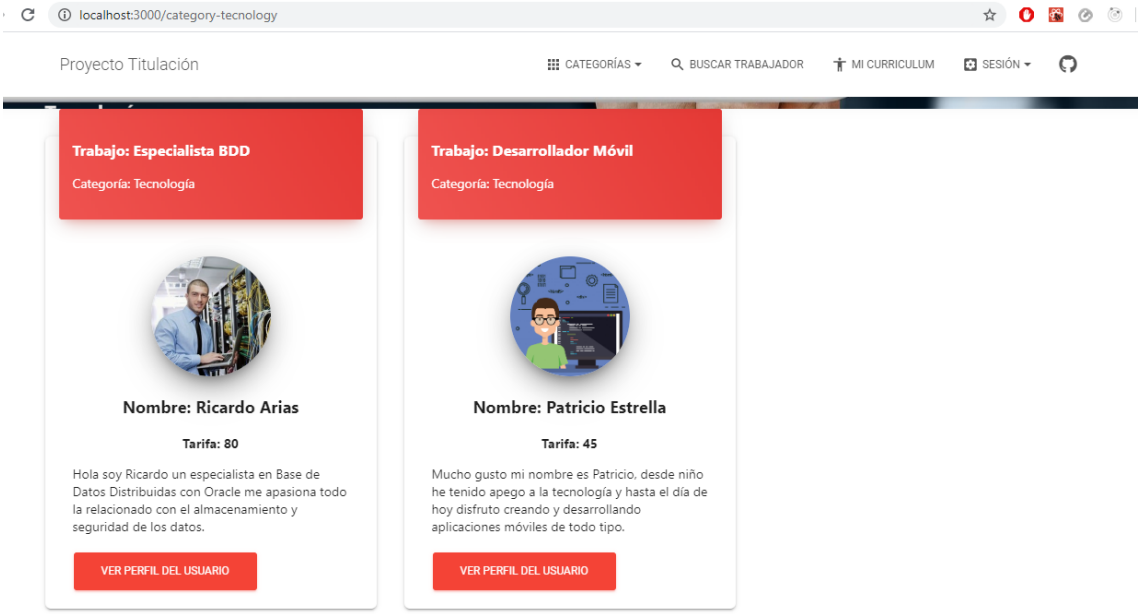


Figura 46. Despliegue de Usuarios con Categoría Tecnología

4.4.6 Sprint Restrospective del Cuarto Sprint

En lo que respecta al sprint restrospective del cuarto sprint se establecen ciertas mejoras a implementar en el proyecto, en este punto del proyecto se considera la implementación de Infinite Scroll para manejar de mejor manera el despliegue de los usuarios a medida que crece la aplicación y así facilitar el uso de la misma, de igual forma se establece mejorar el rendimiento de la aplicación por medio de la integración de Lazy Loading para que el usuario pueda tener un rendimiento óptimo al descargar justamente lo necesario a visualizar.

4.5 Quinto Sprint

A continuación, se redacta el procedimiento que se llevó a cabo para la culminación del quinto sprint.

4.5.1 Sprint Planning

En lo que respecta a la reunión del sprint planning del quinto sprint, se imparten las metas y entregables a implementar en este punto del proyecto. Los componentes por definirse constituyen la funcionalidad de visualizar el perfil de un trabajador en específico, la funcionalidad de contratación y la calificación de la contratación realizada, En primer lugar, el empleador debe ser capaz de acceder al perfil del trabajador deseado y se debe desplegar la información más relevante del curriculum del trabajador, de esta manera se facilita la toma de decisión para la elección y contratación de ese trabajador. Consecuentemente en este sprint es necesario definir la funcionalidad del botón de contratación, se debe definir la lógica para enviar los datos asociados de dicha contratación hacia una nueva API en Amazon Gateway y una nueva tabla en Amazon DynamoDB, adicionalmente se debe establecer un ModalComponent que debe desplegarse indicando que los datos de contratación fueron enviados satisfactoriamente, adicionalmente en el modal debe incluirse

información referente a los datos de contacto asociados a dicho trabajador, finalmente en este sprint se debe implementar la funcionalidad de calificación referente a la puntuación y review de la contratación a criterio del empleador. Para el final del sprint se debe entregar el PerfilTrabajadorComponent que se define a partir de la reutilización de HeaderComponent y FooterComponent, el nuevo componente asociado al perfil del trabajador, el componente ModalComponent según lo definido anteriormente y el componente CalificacionComponent referente a la calificación de la contratación.

4.5.2 Historias de Usuario del Sprint Backlog del Quinto Sprint

Tabla 18.

Historia de Usuario Despliegue de Perfil de Trabajador

Historia de usuario	
ID: CON_01	**Usuario:** Daniel Unapanta
Nombre de historia: Despliegue de Perfil de Trabajador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 5PH	**Sprint asignado:** 5
Responsable: Daniel Unapanta	
Descripción: Como empleador, deseo desplegar el perfil del trabajador que seleccione previamente en categorías, para verificar la información del curriculum del trabajador y facilitar la decisión de contratación.	
Criterios de aceptación: Se debe proporcionar la sección de perfil de trabajador en donde se despliega información relevante de su curriculum: fotografía personal, nombre, tarifa, ciudad, país, acerca de mí y experiencia.	

Tabla 19.

Historia de Usuario Despliegue de Modal Asociado a Contratación

Historia de usuario

ID: CON_02	**Usuario:** Daniel Unapanta
Nombre de historia: Despliegue de Modal Asociado a Contratación	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** N/A
Estimación: 5PH	**Sprint asignado:** 5
Responsable: Daniel Unapanta	
Descripción: Como empleador, deseo visualizar un modal al haber realizado una contratación, para informar la contratación satisfactoria y desplegar los datos de contacto del trabajador	
Criterios de aceptación: Se debe proporcionar un modal que incluya información de la contratación realizada en donde se debe especificar el nombre del trabajador contratado, el trabajo, su correo electrónico y su número de contacto.	

Tabla 20.

Historia de Usuario Ingreso de Calificación al Trabajador por parte del Empleador

Historia de usuario	
ID: CON_03	**Usuario:** Daniel Unapanta
Nombre de historia: Ingreso de Calificación al Trabajador por parte del Empleador	
Prioridad en negocio: Media	**Riesgo en desarrollo:** N/A
Estimación: 3PH	**Sprint asignado:** 5
Responsable: Daniel Unapanta	
Descripción: Como empleador, deseo realizar la calificación correspondiente a la contratación que fue realizada, para puntuar el trabajo elaborado según criterio del empleador.	
Criterios de aceptación: El Empleador al haber realizado la contratación se habilita la sección de calificación, la cual consta del apartado de puntuación de una a cinco estrellas y la sección de review en la que el empleador puede compartir su experiencia con la contratación realizada.	

4.5.3 Sprint Review

Se lleva a cabo la revisión del quinto sprint para analizar y validar el incremento realizado, se añadieron tres nuevos componentes a la aplicación web los cuales se detallan a continuación:

- **PerfilTrabajadorComponent:** Este es el componente encargado de desplegar el perfil del trabajador según los datos definidos en el curriculum previamente. Este componente se define a partir de la reutilización de los componentes HeaderComponent y FooterComponent además se utilizan los componentes de MaterialUI: GridContainer, GridItem, Card, Spinner, Slide. En este componente es necesario realizar la petición a la API de curriculum, obtener los datos del trabajador, el método Storage de Amplify para obtener la fotografía del trabajador y el método Auth de Amplify para obtener los datos del actual empleador. Los datos obtenidos de las peticiones se las almacena en el componente por medio de useState y se hace uso del hook useEffect para hacer el llamado justo después de renderizar los componentes por primera vez.
- **ModalComponent:** Este componente se despliega al haber realizado correctamente una contratación, es decir al haber almacenado los datos de contratación en DynamoDB que se conforman de: id_contrato, id_empleador, id_trabajador, username_trabajador, calificación y review_empleador. Para la definición del componente se hace uso de varios componentes de Material UI como son: Slide, IconButton, Dialog, DialogTitle, DialogContent, DialogActions.
- **CalificaciónComponent:** Este componente se habilita al momento en que el empleador realizó la contratación e implementa la funcionalidad correspondiente a la retroalimentación que proporciona el empleador hacia el trabajador. Para la creación de este componente se hace uso de los componentes de Material UI: Card, CardHeader, CardBody, GridContainer y GridItem, adicionalmente se usa un paquete de npm denominado react-raiting-stars-component

para establecer la funcionalidad de puntuación con una a cinco estrellas la contratación realizada.

4.5.4 Código Fuente del Quinto Sprint

Creación de API relacionada a contratos por medio de la funcionalidad de Amplify add api.

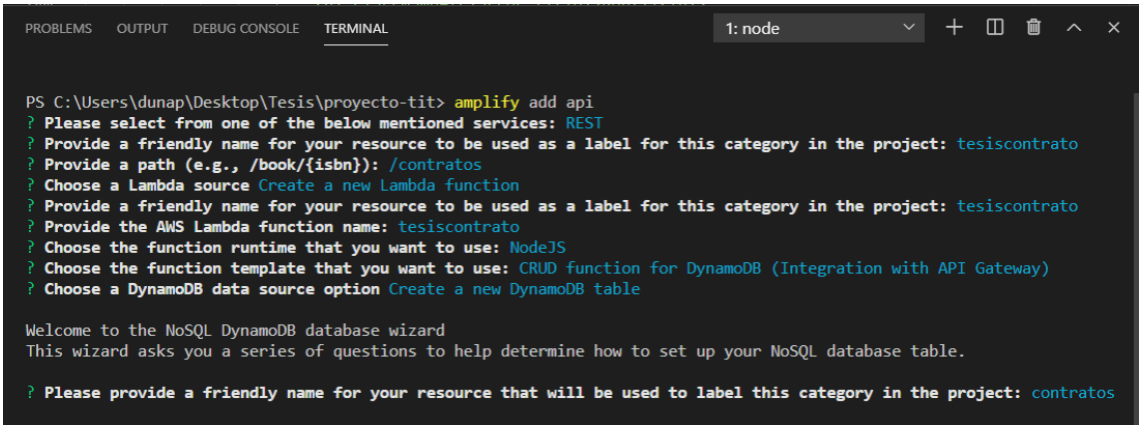


Figura 47. Creación de API Contratos en Amazon API Gateway

Creación de nueva tabla en Amazon DynamoDB para almacenar la información de los contratos realizados, por medio de la funcionalidad de Amplify

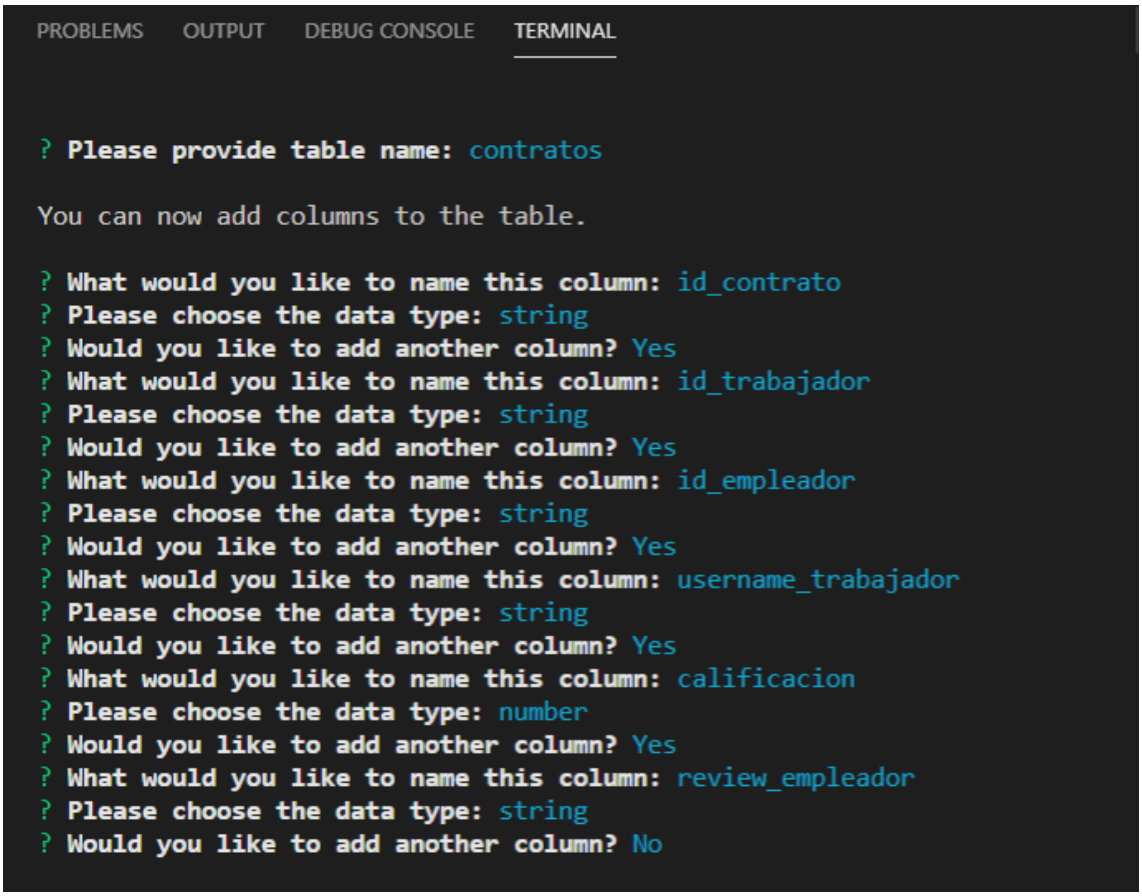


Figura 48.Creación de Tabla Contratos en Amazon Dynamo

Definición de función asíncrona para la obtención de datos de perfil de trabajador y almacenarlos en el estado del componente por medio de la función definida en useState.

```
const getUserCurriculumAsync = async () => {
    let path = `/users/${location.state.id_trabajador}`;
    const apiName = "tesis";

    const response  = await API.get(apiName, path);
    console.log("respuestita", response)
    try{
    setUserCurriculum({
        firstName:response[0].firstName,
        lastName:response[0].lastName,
        cedula:response[0].cedula,
        telefono:response[0].telefono,
        categoria:response[0].categoria,
        trabajo:response[0].trabajo,
        tarifa:response[0].tarifa,
        ciudad:response[0].ciudad,
        pais:response[0].pais,
        postalCode:response[0].postalCode,
        aboutMe:response[0].aboutMe,
        experiencia:response[0].experiencia,
        user:response[0].user,
        })
        console.log("Final usuario con useState",userCurriculum)}
        catch(err){
            console.log("Error que se prodria mejorar mas adelante",err)
        }
```

Figura 49.Función asíncrona de Perfil Trabajador

Definición de método asíncrono para el envío de los datos de contratación hacia DynamoDB y despliegue de modal correspondiente a los datos de contacto de trabajador.

```
const submitCurriculumAsync = async () => {
    await getCurrentUserAsync()
    let apiName = "tesiscontrato";
    let path = "/contratos";
    let data = {
        body: {
            id_contrato: uuid(),
            id_empleador: empleador.user.sub,
            id_trabajador: userCurriculum.user.sub,
            username_trabajador: userCurriculum.user.username,
            calificacion: "",
            review_empleador: ""
        }
    };
    console.log("Contrato a dynamodb")
    await API.post(apiName, path, data)
    console.log("Enviado")
    setClassicModal(true)
}
```

Figura 50. Función Asíncrona para Envió de Datos de Contratación

Definición de componente modal por medio de componentes de Material UI y estructura de datos de contacto de trabajador.

```
</DialogTitle>
<DialogContent
    id="classic-modal-slide-description"
    className={classes.modalBody}
>
    <p>
        Has Contratado a <strong>{userCurriculum.firstName}
        {userCurriculum.lastName}</strong> por su
        trabajo: <strong>{userCurriculum.trabajo}</strong>. Ahora puedes
        ponerte en contacto con el trabajador
        con su email <strong>{userCurriculum.user.email}</strong> o
        número célular <strong>{userCurriculum.user.phone_number}</
        strong>
    </p>
    <p>
        Gracias por tu Contratación :)
    </p>
</DialogContent>
<DialogActions className={classes.modalFooter}>
    <Button
        onClick={() => setClassicModal(false)}
        color="success"
        >
        Aceptar
    </Button>
</DialogActions>
```

Figura 51. Definición de Modal con Datos de Contacto de Trabajador

Definición de método asíncrono para él envió de la calificación asignada a la contratación hacia DynamoDB.

```
const submitCalificacion = async () => {
    let apiName = "tesiscontrato";
    let path = "/contratos";
    let data = {...datosContrato};
    data.body.calificacion = datosCalificacion.calificacion;
    data.body.review_empleador = datosCalificacion.review_empleador;
    console.log("USESTATE",datosContrato)
    return await API.put(apiName, path, data);
}
```

Figura 52. Función Asíncrona de Envió de Calificación Asignada a DynamoDB

4.5.5 Entregables del Quinto Sprint

A continuación, se presenta una breve descripción de los entregables del quinto sprint:

Como primer entregable del quinto sprint se presenta la actualización del HeaderComponente que se despliega en Perfil Trabajador.

Figura 53. Actualización Header Component en PerfilTrabajador

Despliegue del perfil trabajador según los datos obtenidos de la petición a la API curriculum

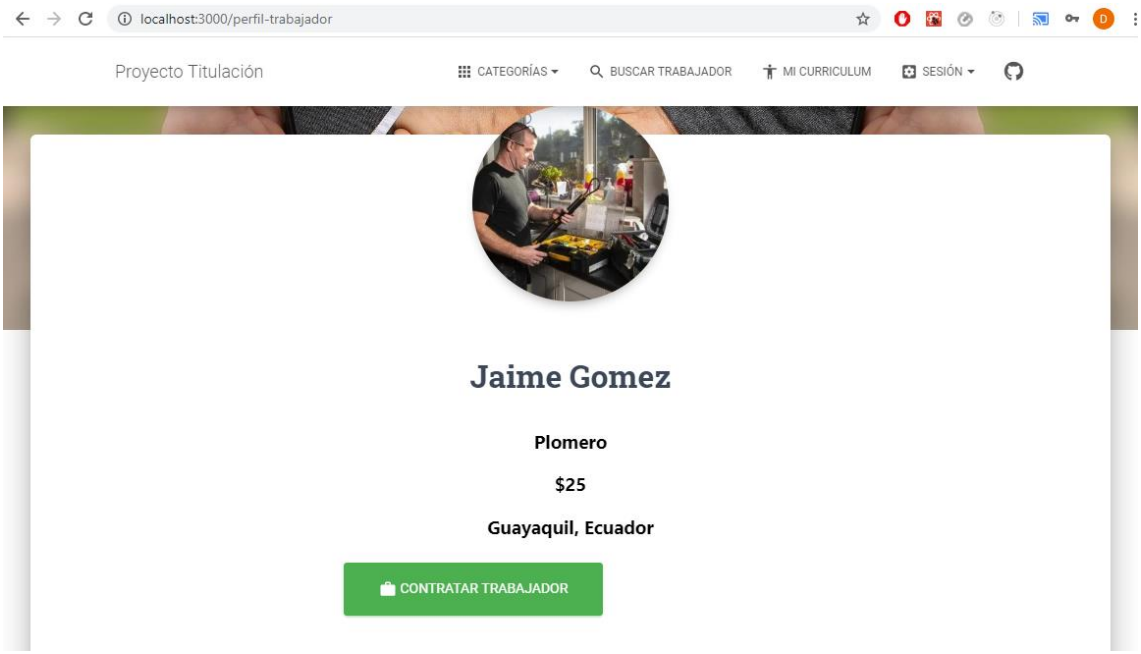


Figura 54. Despliegue Perfil Trabajador Parte 1

Figura 55. Despliegue Perfil Trabajador Parte 2

Despliegue del modal donde se especifica la contratación satisfactoria del trabajador y los datos de contacto de dicho trabajador.

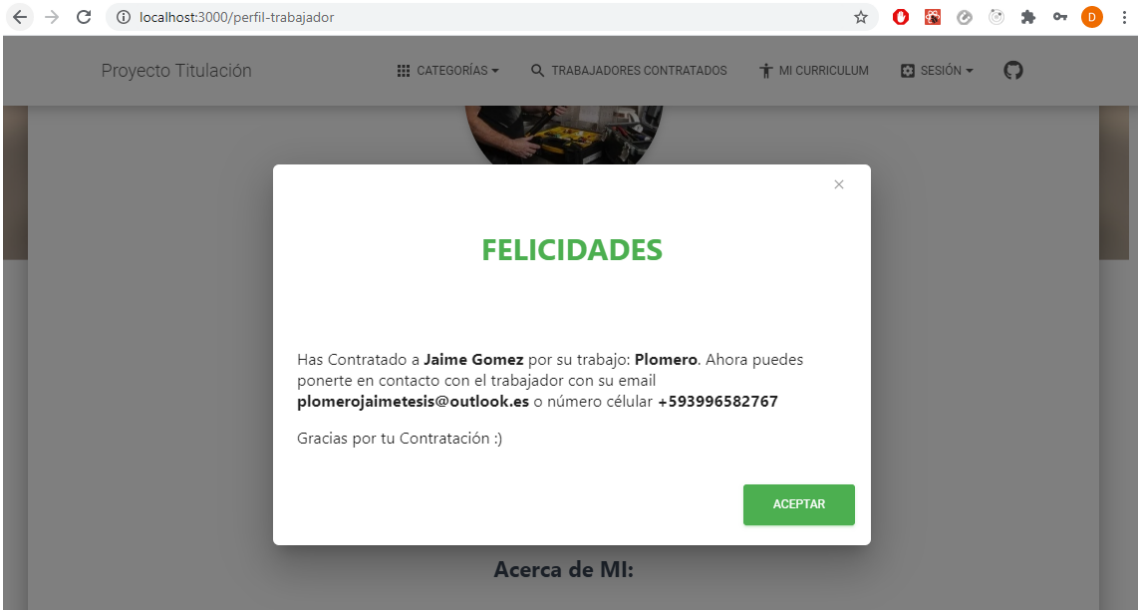


Figura 56. Despliegue de Modal Referente a Contratación

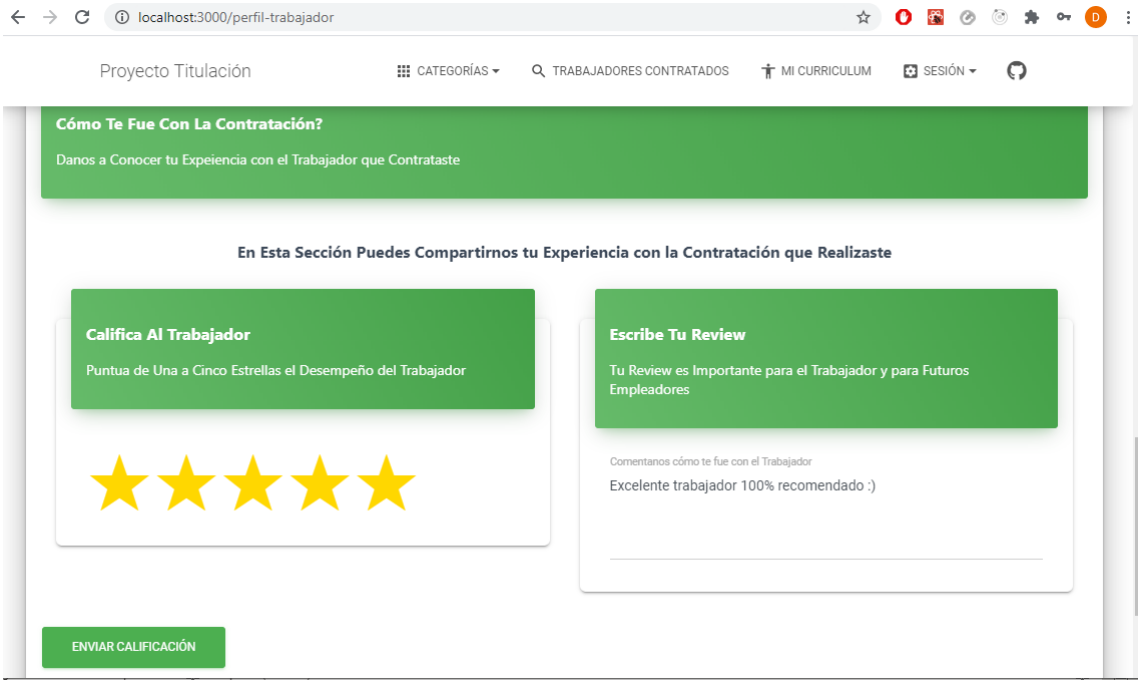


Figura 57. Despliegue de Componente Calificación de Contratación

Verificación de almacenamiento de contrato que consta de los campos: calificación, fecha_contratacion, id_contrato, id_empleador_id_trabajador, review_empleador, user_trabajador.

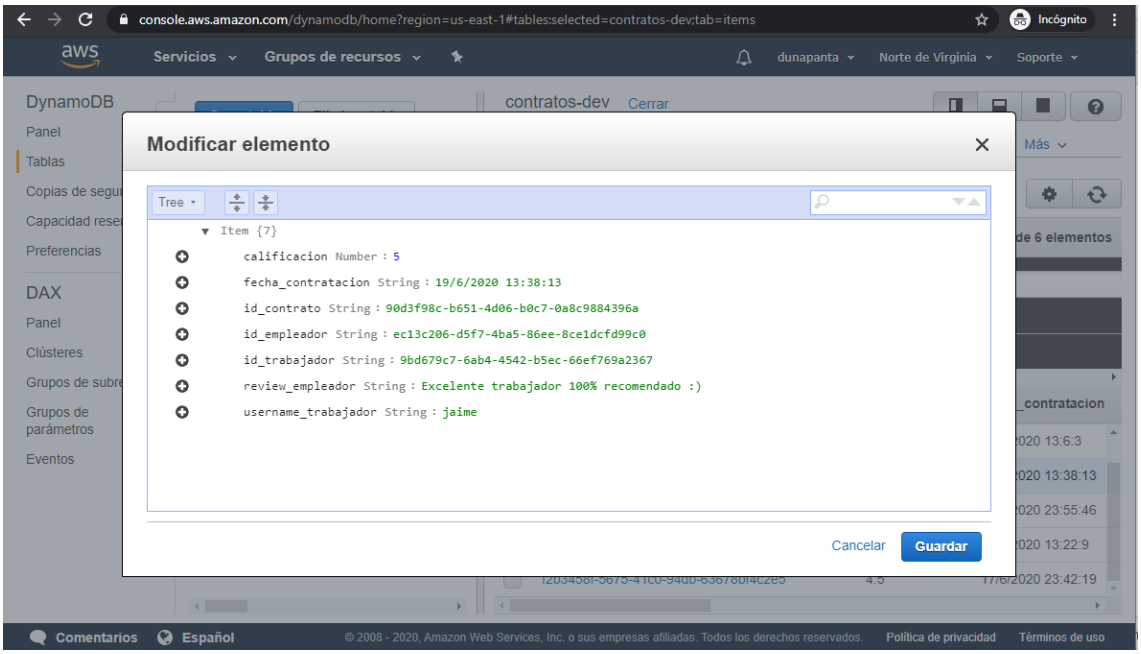


Figura 58. Verificación de Almacenamiento de Contrato en DynamoDB

4.5.6 Sprint Restrospective del Quinto Sprint

En lo que respecta al sprint restrospective del quinto se establecen ciertas mejoras a implementar en el proyecto, se considera la definición de pruebas unitarias con jest, para la verificación de funcionalidad de varios componentes que conforman el sprint y así asegurarse que se comportan correctamente.

4.6 Sexto Sprint

4.6.1 Sprint Planning

En lo acorde a la reunión del sprint planning del sexto sprint, se establecen las metas y entregables a implementar para la finalización del proyecto. Los componentes por definirse constituyen la funcionalidad del despliegue

de los trabajadores contratados, es decir la lista con información relevante acerca de cada una de las contrataciones realizadas, en este caso se debe presentar al usuario una tabla con cinco columnas las cuales constan: nombre del trabajador, la categoría a la que pertenece, el trabajo que realiza, la fecha y hora en que se realizó la contratación y la calificación asignada por el empleador. Para la finalización del sprint se debe presentar un nuevo componente denominado ListaContratosComponent que contenga la lógica de despliegue de la tabla mencionada.

4.6.2 Historias de Usuario del Sprint Backlog del Sexto Sprint

Tabla 21.

Historia de Usuario Despliegue de Lista de Todos los Trabajadores Contratados

Historia de usuario	
ID: LIS_01	**Usuario:** Daniel Unapanta
Nombre de historia: Despliegue de Lista de Todos los Trabajadores Contratados	
Prioridad en negocio: Media	**Riesgo en desarrollo:** N/A
Estimación: 5PH	**Sprint asignado:** 6
Responsable: Daniel Unapanta	
Descripción: Como empleador, deseo desplegar la lista de todos los trabajadores que he contratado, para verificar el registro de contrataciones realizadas.	
Criterios de aceptación: El Empleador tiene la opción de visualizar el listado de todas las contrataciones realizadas, en donde se detalla el nombre del trabajador contratado, la categoría, el trabajo, la fecha de contratación y la calificación asignada a dicha contratación.	

4.6.3 Sprint Review

Se lleva a cabo la revisión del sexto sprint para analizar y validar el incremento realizado, se añadió un nuevo componente a la aplicación web el cual se detalla a continuación:

- **CalificaciónComponent:** Este componente es el encargado de desplegar la lista de todos los trabajadores contratados según la definición de la tabla establecida. Para la creación de este componente se hace uso de los componentes de Material UI: Table, TableHead, TableRow, TableBody, TableCell, Button, Card, CardHeader, CardBody, GridContainer y GridItem, y se reutiliza el componente react-raiting-stars-component establecido en el quinto sprint para desplegar la puntuación de una a cinco estrellas asignada previamente por el empleador, en caso que el empleador no haya asignado alguna calificación, se renderiza un botón que permite al empleador navegar al perfil del empleado para poder realizar la calificación de la contratación.

4.6.4 Código Fuente del Sexto Sprint

Definición de método asíncrono para obtención de lista trabajadores contratados y asignación de campos relevantes en cada uno de los objetos obtenidos.

```
for ( let contratado of contratosEmpleador) {
    let path =  `/users/${contratado.id_trabajador}`;
    const apiName = "tesis";
    const datosTrabajador  = await API.get(apiName, path);

    datosTrabajador.forEach( (objetoTrab) => {
        objetoTrab.id_contrato = contratado.id_contrato
        objetoTrab.fecha_contratacion = contratado.fecha_contratacion
        objetoTrab.calificacion = contratado.calificacion
    })
    contratados.push(datosTrabajador);
    console.log("Datos TRABAJADOR CONTRATADO", contratados)
}

setDatosContratados(contratados)
```

Figura 59. Función Asíncrona Obtención de Contratos Realizados

Definición de estructura de los datos de la tabla y asignación de cada uno de los campos correspondientes a las contrataciones realizadas

```
let listaAux = []
contratados.forEach( async (trab) => {
    trab.forEach( async (traba) => {
        listaAux.push([`${traba.firstName} ${traba.lastName}`, traba.
        categoria, traba.trabajo, traba.fecha_contratacion, traba.
        calificacion ? <ReactStars
        size={25}
        count={5}
        value={+traba.calificacion}
       /*  onChange={ratingChanged} */
        color2={"#ffd700"}
    /> : <Button color="info" onClick={() => handlePerfil(traba.user_id,
    traba.user.username, traba.id_contrato)}><Create className={classes.
    icon} />  Calificar</Button>])
    } )
} )
```

Figura 60. Función de Asignación de Campos en la Tabla

Renderizado de la tabla según los campos establecidos: nombre trabajador, categoría, trabajo, fecha de contratación y calificación asignada.

```
<TableBody>
  {tableData.map((prop, key) => {
    return (
      <TableRow key={key} className={classes.tableBodyRow}>
        {prop.map((prop, key) => {
          return (
            <TableCell className={classes.tableCell} key={key}>
              {prop}
            </TableCell>
          );
        })}
      </TableRow>
    );
  })}
</TableBody>
```

Figura 61.Renderizado de Campos en la Tabla

4.6.5 Entregables del Sexto Sprint

A continuación, se presenta la descripción correspondiente a los entregables del sexto sprint:

Como primer entregable del sexto sprint se presenta la actualización del HeaderComponente que se despliega Lista de Contratos

Figura 62. Actualización Header Component en ListaContratos

Despliegue de tabla correspondiente al registro de contrataciones realizadas

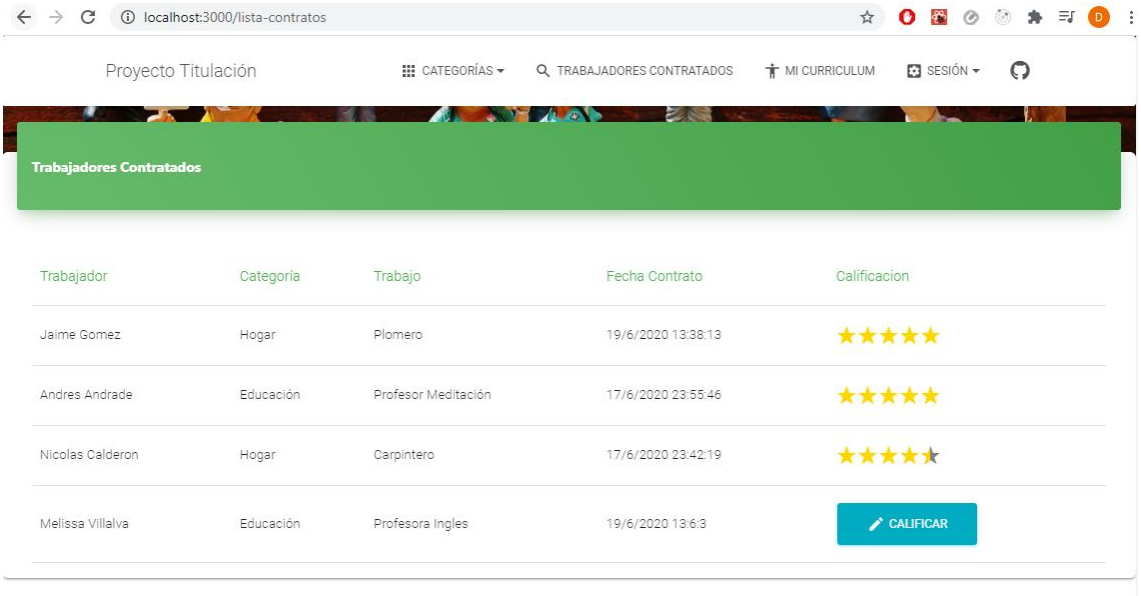


Figura 63. Tabla de Lista de Contrataciones Realizadas

4.6.6 Sprint Restrospective del Sexto Sprint

En lo que respecta al sprint restrospective del sexto sprint se establecen ciertas mejoras a implementar en el proyecto, se considera la redefinición de pruebas unitarias con vez haciendo uso de enzyme, librería de testing enfocada a React ya que facilita realización de las pruebas unitarias en los componentes de la aplicación.

5 Capítulo V: Pruebas Unitarias y Casos de Prueba de la Aplicación Web

En el presente capítulo se presentan las pruebas correspondientes a cada uno de los sprint realizados y los resultados finales obtenidos al haber concluido el proyecto.

Para la implementación de las pruebas unitarias en este capítulo, se hace uso de tres herramientas enfocadas a testing:

- **Jest:** Es un framework de testing para Javascript con la habilidad de adaptarse a cualquier librería o framework de Javascript, Jest permite la ejecución rápida y paralela de pruebas. Esta incluido por defecto con Create React App y se lo inicializa en modo de observación con el comando npm run test. (Bulat, 2019)
- **Enzyme:** Es un paquete de utilidad de testing desarrollado por Airbnb que facilita el testing de componentes en React, Enzyme proporciona métodos más optimizados para llevar a cabo las pruebas. (Bulat, 2019)
- **Enzyme-to-json:** Es un paquete de npm enfocado a convertir los wrappers de Enzyme a un formato compatible con los snapshots de Jest. (Adrien, s.f.)

5.1 Pruebas del Primer Sprint

A continuación, se presentan las pruebas realizadas para el primer Sprint

5.1.1 Prueba Unitaria del Primer Sprint

A continuación, se presenta la prueba del renderizado correcto del componente padre referente a Autenticación.

Se define el test por medio de la utilización del método shallow definido en enzyme y se lo compara con un snapshot del componente Authentication para cerciorarse que el renderizado fue satisfactorio.

```
import React from 'react';
import { shallow } from 'enzyme';
import Authentication from 'views/Auth/Authentication'

describe('Pruebas en Componente de Autenticacion', () => {

    test('Debe mostrar Autenticación correctamente', () =>{
        const wrapper = shallow(<Authentication />)
        expect( wrapper ).toMatchSnapshot();
    })

})
```

Figura 64. Definición de Prueba Unitaria de Renderizado de Autenticación

A continuación, se presenta el resultado obtenido de la prueba unitaria definida anteriormente con un resultado satisfactorio

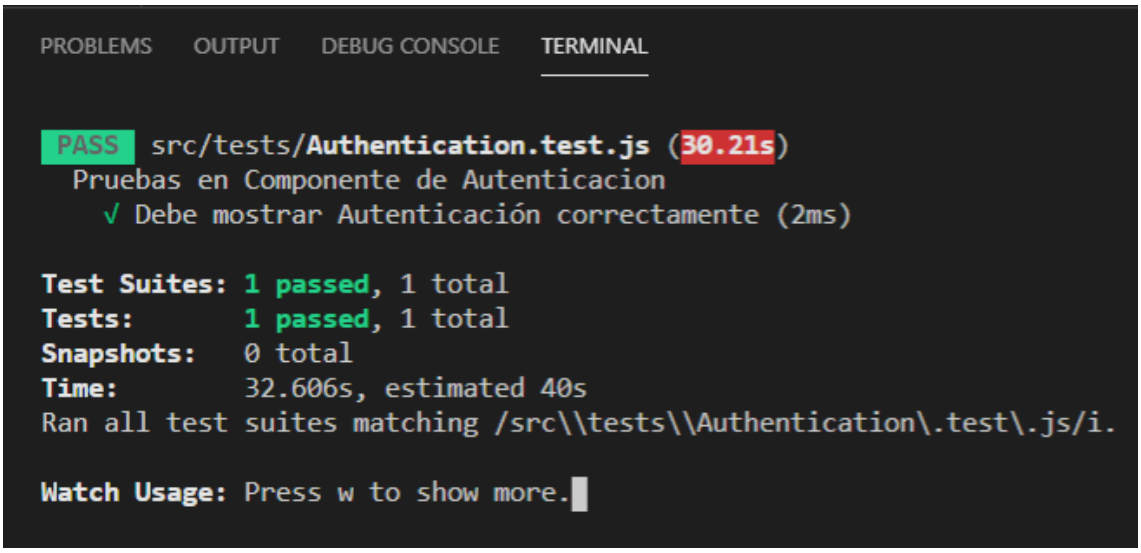


Figura 65. Prueba Unitaria de Renderizado de Autenticación Satisfactoria

5.1.2 Caso de Prueba del Primer Sprint

A continuación, se presenta el caso de prueba para el sprint 1 referente a la creación de una nueva cuenta en la aplicación web.

Tabla 22.

Caso de Uso Creación de una nueva cuenta

Caso de Prueba	
ID: CAS_01	**Fecha:** 26-Junio-2020
Nombre de historia: Creación de una nueva cuenta	
Descripción: Creación de una nueva cuenta en la aplicación web mediante el correo electrónico dunapanta@outllok.es seguir paso a paso los requerimientos para la creación de una nueva cuenta y verificar en Amazon Cognito la cuenta creada.	
Resultado:	

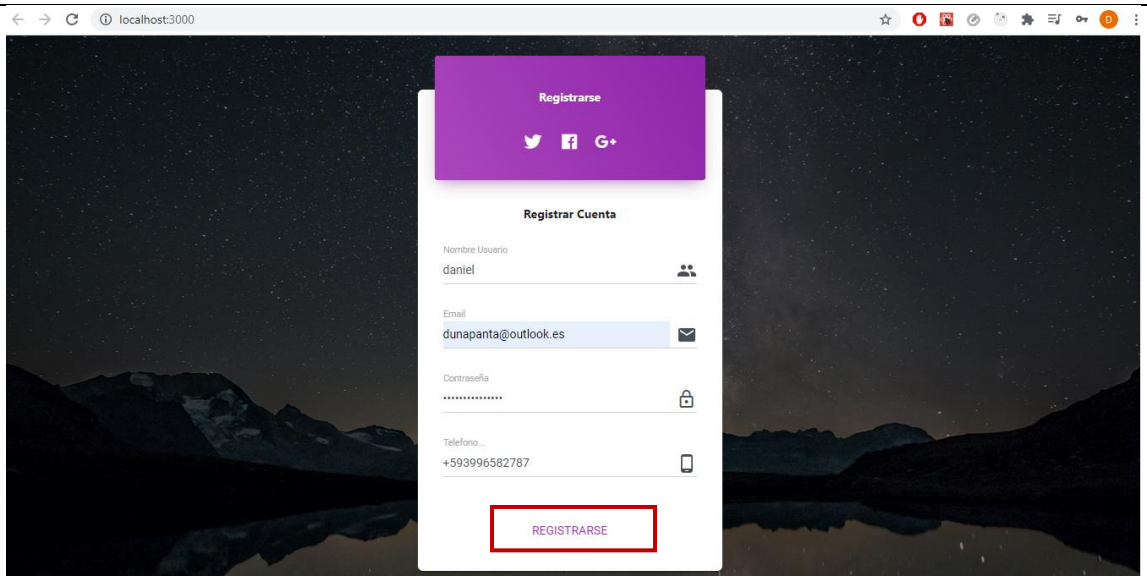


Figura 66. Resultado Caso de Prueba Uno Ingreso de Registro

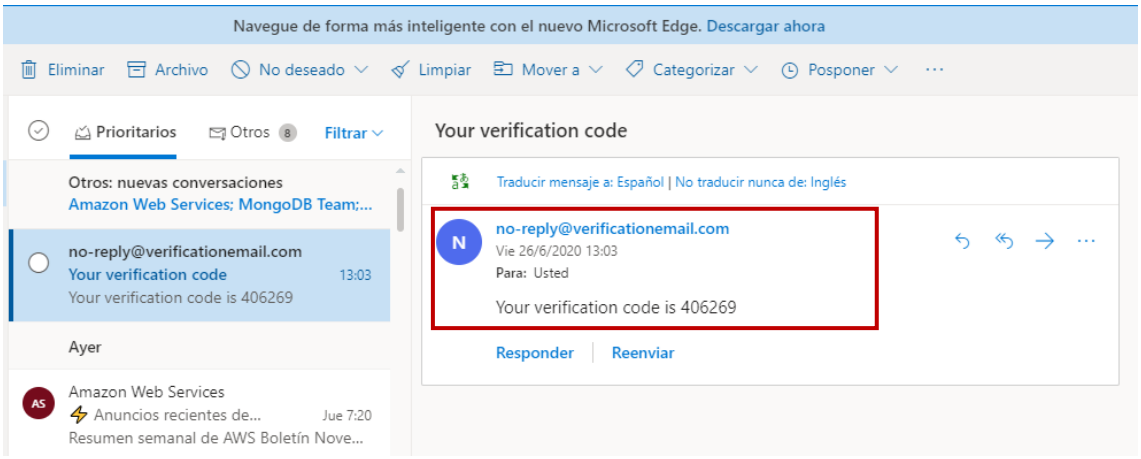


Figura 67. Resultado Caso de Prueba Uno Envió de Código de Verificación a Correo Electrónico

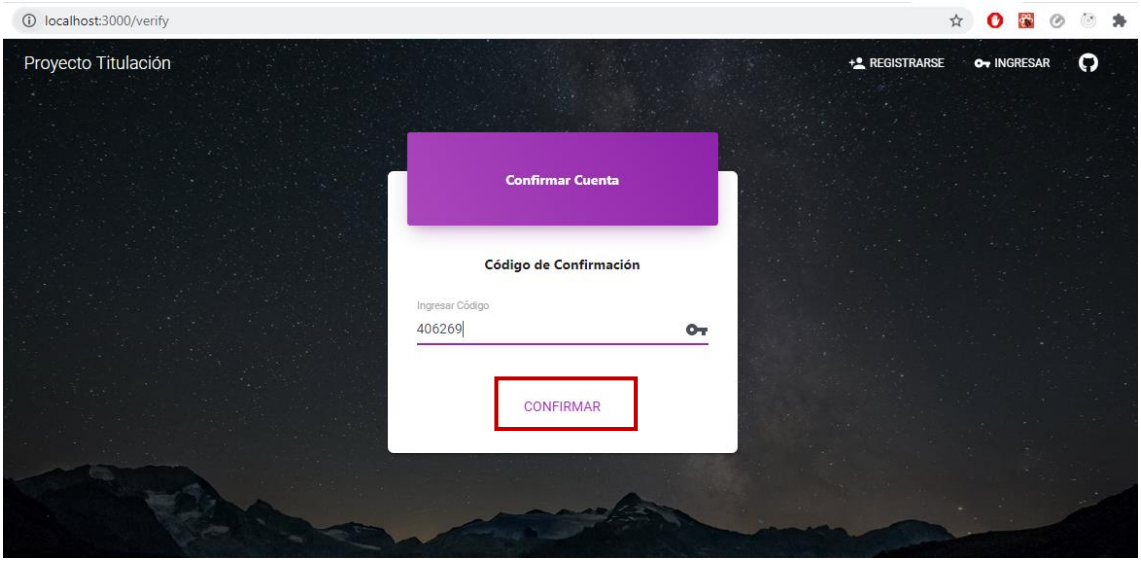


Figura 68. Resultado Caso de Prueba Uno Ingreso de Código de Verificación

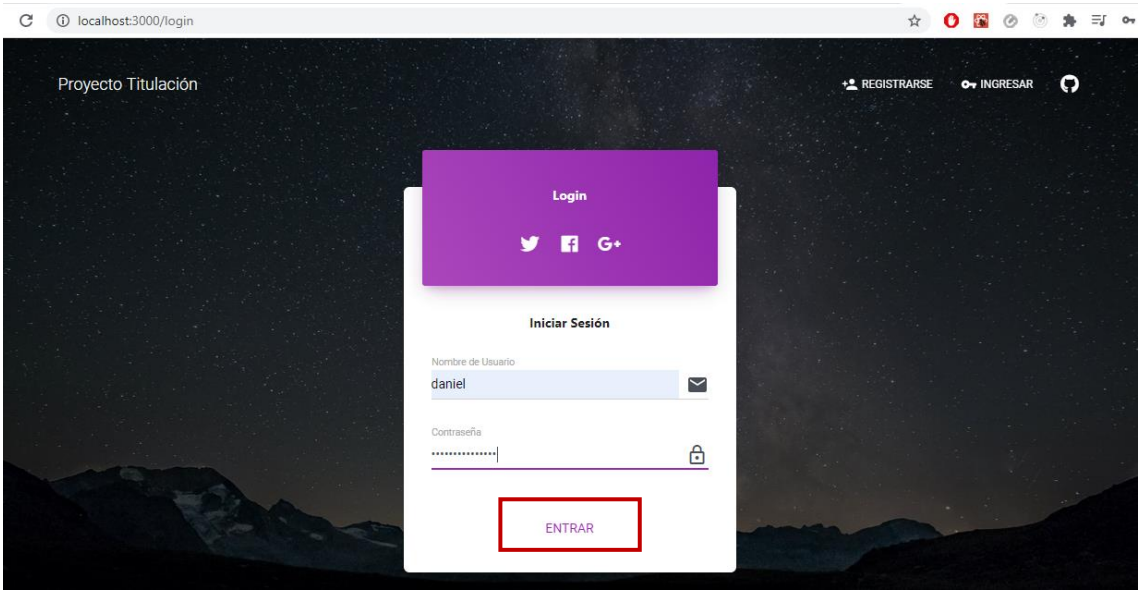


Figura 69. Resultado Caso de Prueba Uno Inicio de Sesión

Figura 70. Resultado Caso de Prueba Uno Verificación de Cuenta Creada en Amazon Cognito

5.2 Pruebas del Segundo Sprint

A continuación, se presentan las pruebas realizadas para el segundo Sprint

5.2.1 Prueba Unitaria del Segundo Sprint

Se exhibe la prueba del renderizado correspondiente al componente padre de Home.

De manera similar se define la prueba por medio de la utilización del método shallow de enzyme y se lo compara con un snapshot del componente HomePage para cerciorarse que el renderizado fue correcto.

```
import React from 'react';
import { shallow } from 'enzyme';
import HomePage from 'views/Home/HomePage'

describe('Pruebas en Componente de Autenticacion', () => {

    test('Debe mostrar Autenticación correctamente', () =>{
        const wrapper = shallow(<HomePage />)
        expect( wrapper ).toMatchSnapshot();
    })

})
```

Figura 71. Definición de Prueba Unitaria de Renderizado de HomePage

A continuación, se presenta el resultado satisfactorio de la prueba obtenida previamente

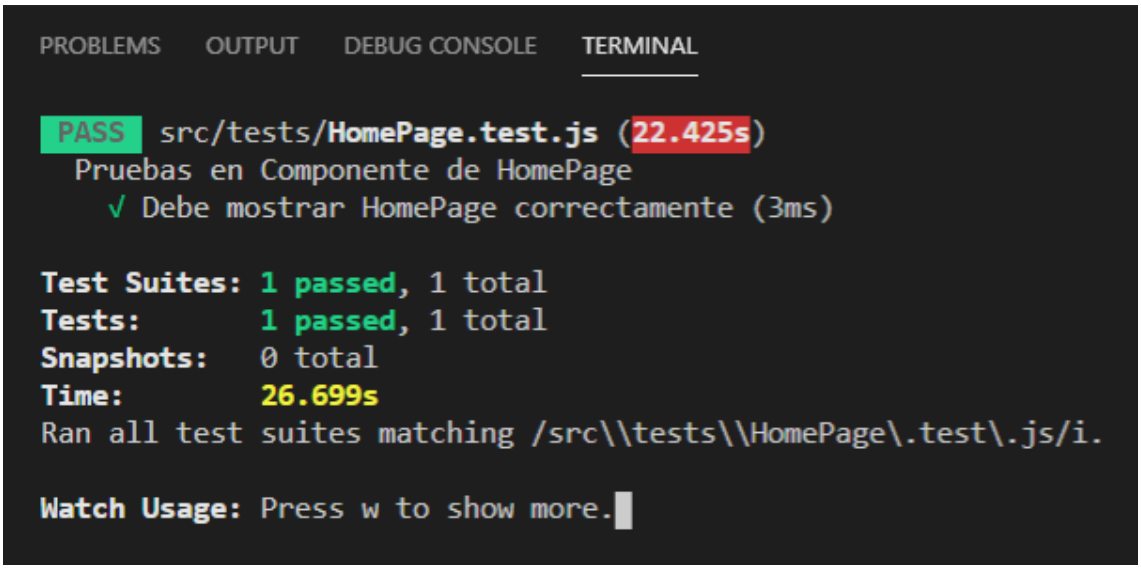


Figura 72. Prueba Unitaria de Renderizado de HomePage Satisfactoria

5.2.2 Caso de Prueba del Segundo Sprint

A continuación, se presenta el caso de prueba del segundo sprint correspondiente al acceso de la Home Page de la aplicación web.

Tabla 23.

Caso de Uso Acceso a Home Page de la Aplicación Web

Caso de Prueba	
ID: CAS_02	**Fecha:** 26-Junio-2020
Nombre de historia: Acceso a Home Page de la Aplicación Web	
Descripción: Acceder a la Home Page de la aplicación web al haber iniciado sesión correctamente y verificar el despliegue de componentes asociados.	
Resultado:	

Figura 73. Resultado Caso de Prueba Dos Acceso a la Home Page

5.3 Pruebas del Tercer Sprint

A continuación, se presentan las pruebas realizadas para el tercer Sprint

5.3.1 Pruebas Unitarias del Tercer Sprint

A continuación, se define la prueba del renderizado correspondiente al componente padre de UserCurriculum.

De manera similar se define la prueba por medio de la utilización del método shallow de enzyme y se lo compara con un snapshot del componente Curriculum para cerciorarse que el renderizado fue correcto.

```
import React from 'react';
import { shallow } from 'enzyme';
import UserCurriculum from 'views/UserCurriculum/UserCurriculum'

describe('Pruebas en Componente de UserCurriculum', () => {

    test('Debe mostrar UserCurriculum correctamente', () =>{
        const wrapper = shallow(<UserCurriculum />)
        expect( wrapper ).toMatchSnapshot();
    })

})
```

Figura 74. Definición de Prueba Unitaria de Renderizado de UserCurriculum

A continuación, se presenta el resultado satisfactorio de la prueba definida previamente

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

 PASS  src/tests/Curriculum.test.js (24.595s)
  Pruebas en Componente de UserCurriculum
    √ Debe mostrar UserCurriculum correctamente (2ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        24.703s
Ran all test suites matching /src\\tests\\Curriculum\.test\.js/i.

Watch Usage: Press w to show more.
```

Figura 75. Prueba Unitaria de Renderizado de UserCurriculum Satisfactoria

5.3.2 Caso de Prueba del Tercer Sprint

Tabla 24.

Caso de Uso Creación de Curriculum de Trabajador

Caso de Prueba	
ID: CAS_03	**Fecha:** 26-Junio-2020
Nombre de historia: Creación de Curriculum de Trabajador.	
Descripción: Creación de un nuevo curriculum de trabajador mediante los siguientes datos: • Nombre: Daniel • Apellido: Unapanta • Cédula: 1715641468 • Telefono:099658645 • Categoría Trabajo: Tecnología • Nombre Trabajo: Desarrollador Backend • Tarifa: 48 • Ciudad: Quito • País: Ecuador • Codigo Postal: 170417 • Acerca de Mi: Hola mi nombre es Daniel y me apasiona el mundo de la tecnología especialmente backend del desarrollo web. • Mi Experiencia: Mi experiencia consiste en el desarrollo de diferentes proyectos del lado del backend ya sea con Node trabajando con Express o con Python trabajando con Django. Adicionalmente subir una imagen y verificar el almacenamiento de los datos en Amazon DynamoDB y la fotografía en Amazon S3	
Resultado:	

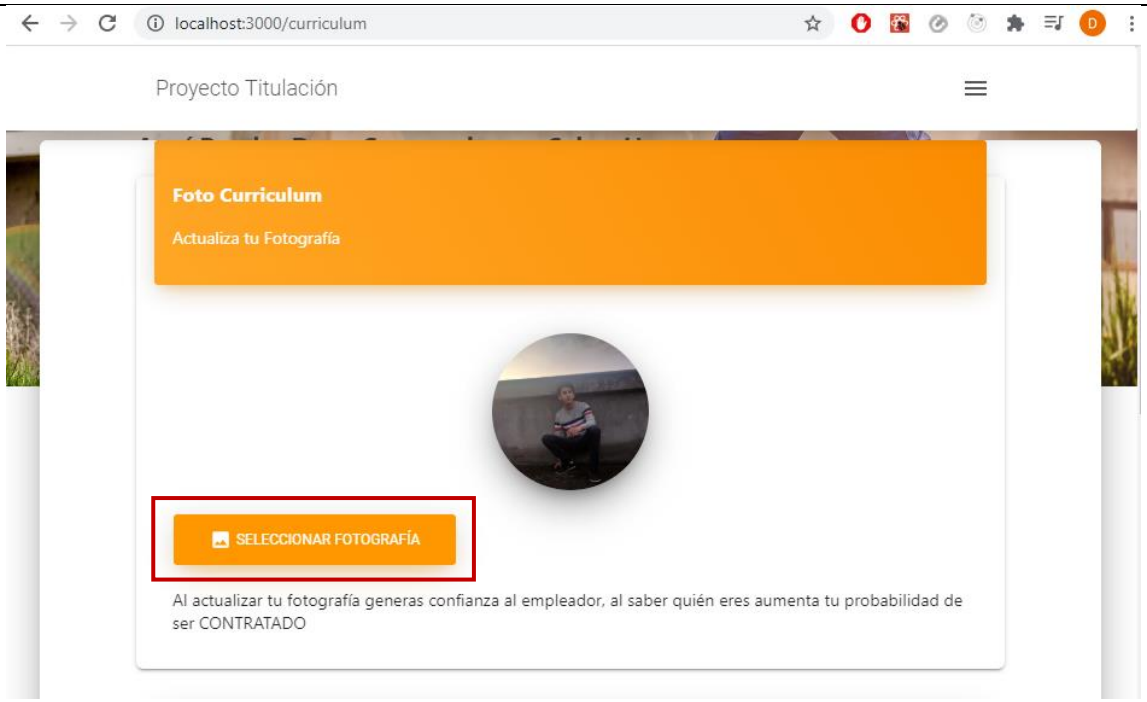


Figura 76. Resultado Caso de Prueba Tres Agregar Fotografía a Curriculum

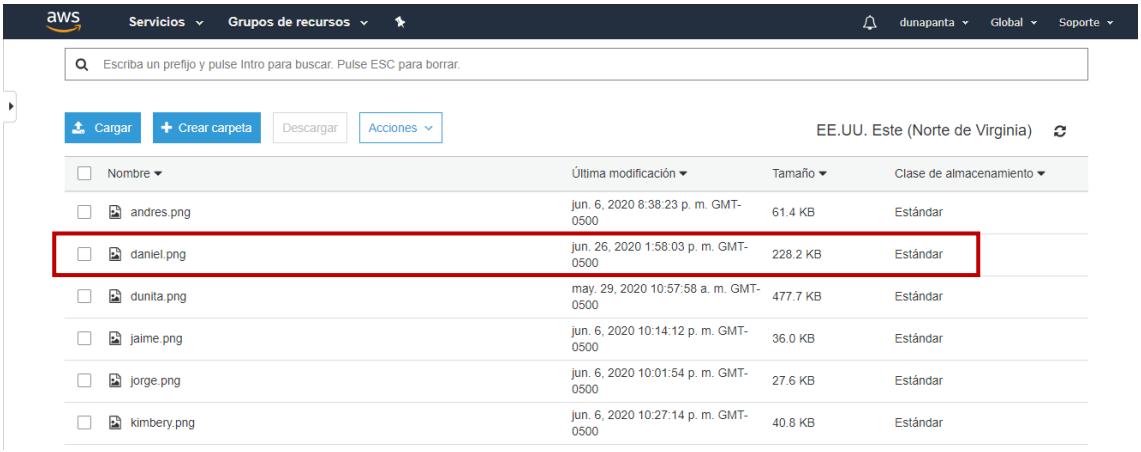


Figura 77. Resultado Caso de Prueba Tres Verificación de Almacenamiento de Fotografía en Amazon S3

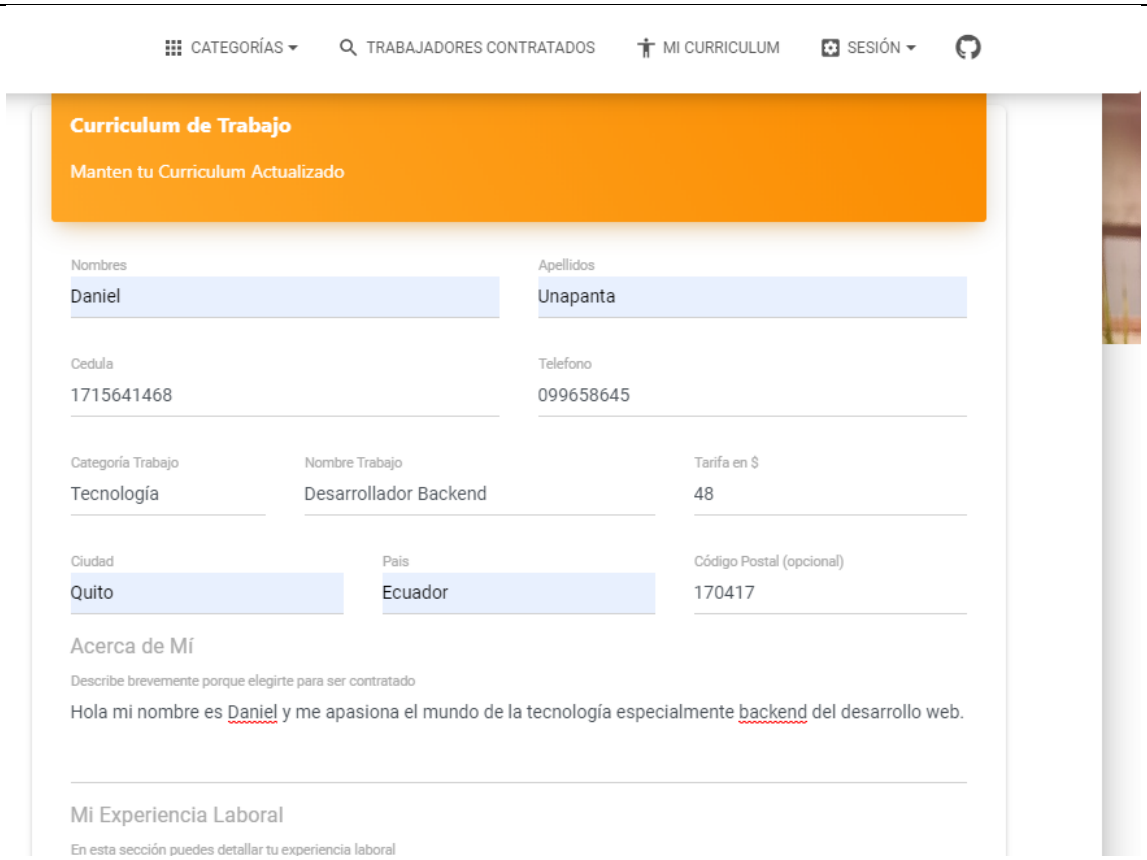


Figura 78. Resultado Caso de Prueba Tres Agregar Datos de Curriculum

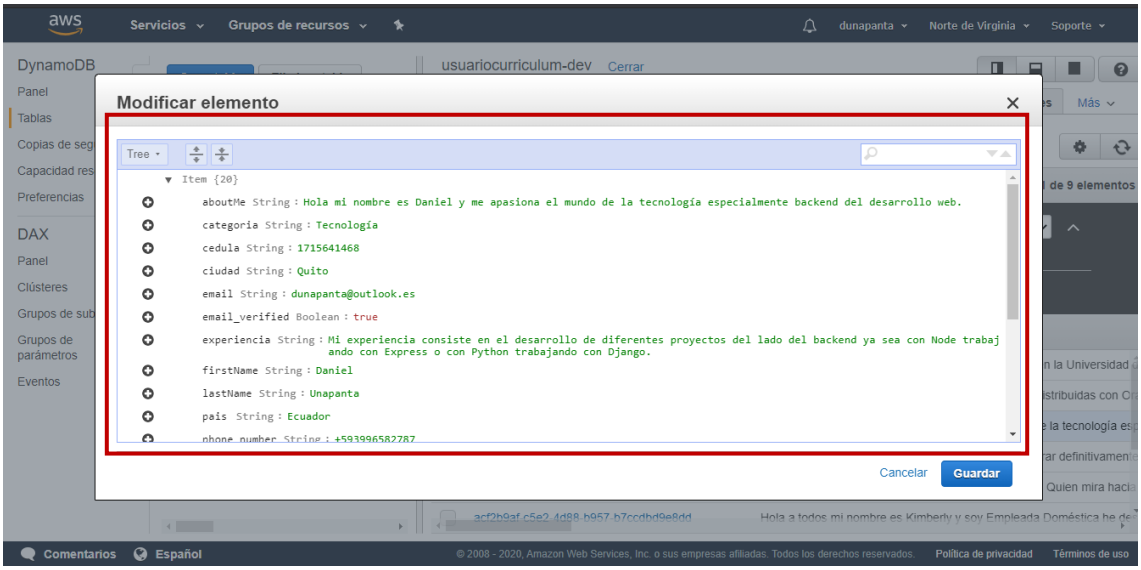


Figura 79. Resultado Caso de Prueba Tres Verificación de Almacenamiento de Datos de Curriculum en Amazon DynamoDB

5.4 Pruebas del Cuarto Sprint

A continuación, se presentan las pruebas realizadas para el cuarto Sprint

5.4.1 Prueba Unitaria del Cuarto Sprint

Se presenta la prueba del renderizado correspondiente al componente padre de Categories.

De igual manera se define la prueba por medio de la utilización del método shallow de enzyme y se lo compara con un snapshot del componente Categories para cerciorarse que el renderizado fue correcto.

```
import React from 'react';
import { shallow } from 'enzyme';
import Categories from 'views/Categories/Categories'

describe('Pruebas en Componente de Categories', () => {

    test('Debe mostrar Categories correctamente', () =>{
        const wrapper = shallow(<Categories />)
        expect( wrapper ).toMatchSnapshot();
    })

})
```

Figura 80. Definición de Prueba Unitaria de Renderizado de Categories

A continuación, se presenta el resultado satisfactorio de la prueba obtenida previamente

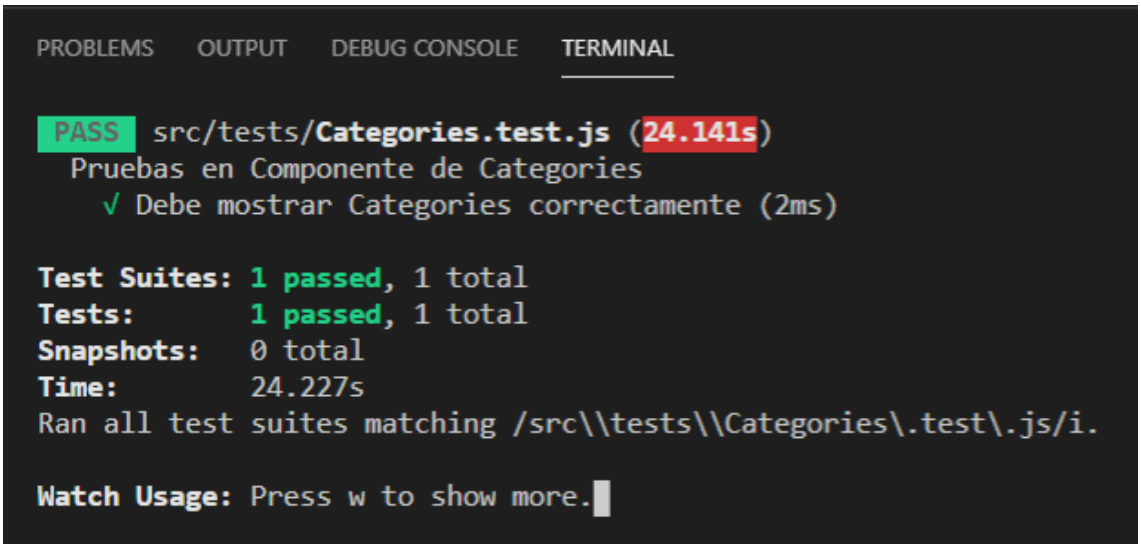


Figura 81. Prueba Unitaria de Renderizado de Categories Satisfactoria

5.4.2 Caso de Prueba del Cuarto Sprint

Tabla 25.

Caso de Uso Despliegue de Trabajadores

Caso de Prueba	
ID: CAS_04	**Fecha:** 26-Junio-2020
Nombre de historia: Despliegue de Trabajadores	
Descripción: Acceder a la sección de categorías de la aplicación y desplegar los trabajadores disponibles para ser contratados.	
Resultado:	

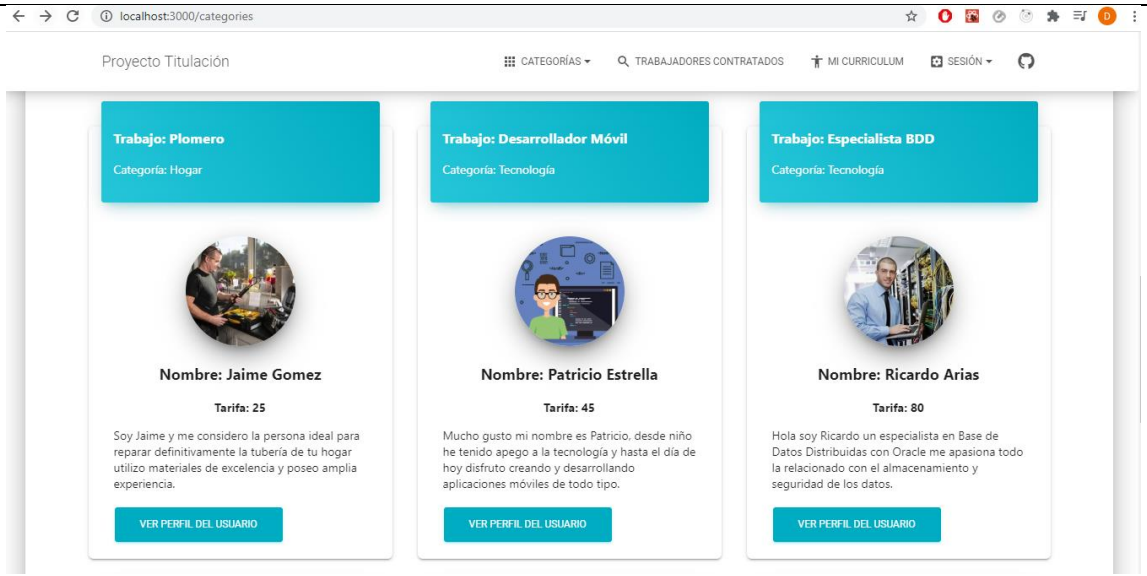


Figura 82. Resultado Caso de Prueba Cuatro Despliegue de Trabajadores

5.5 Pruebas del Quinto Sprint

A continuación, se presentan las pruebas realizadas para el quinto Sprint

5.5.1 Prueba Unitaria del Quinto Sprint

Se presenta la prueba del renderizado correspondiente al componente padre de PerfilTrabajador.

Como se ha venido trabajando se define la prueba por medio de la utilización del método shallow de enzyme y se lo compara con un snapshot del componente PerfilTrabajador para cerciorarse que el renderizado fue correcto.

```
import React from 'react';
import { shallow } from 'enzyme';
import PerfilTrabajador from 'views/PerfilTrabajador/PerfilTrabajador'

describe('Pruebas en Componente de PerfilTrabajador', () => {

    test('Debe mostrar PerfilTrabajador correctamente', () =>{
        const wrapper = shallow(<PerfilTrabajador />)
        expect( wrapper ).toMatchSnapshot();
    })

})
```

Figura 83. Definición de Prueba Unitaria de Renderizado de PerfilTrabajador

A continuación, se presenta el resultado satisfactorio de la prueba obtenida previamente

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

 PASS  src/tests/PerfilTrabajador.test.js (24.698s)
  Pruebas en Componente de PerfilTrabajador
    √ Debe mostrar PerfilTrabajador correctamente (2ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        24.782s
Ran all test suites matching /src\\tests\\PerfilTrabajador\.test\.js/i.

Watch Usage: Press w to show more.
```

Figura 84. Prueba Unitaria de Renderizado de PerfilTrabajador Satisfactoria

5.5.2 Caso de Prueba del Quinto Sprint

Tabla 26.

Caso de Uso Realización de Contratación

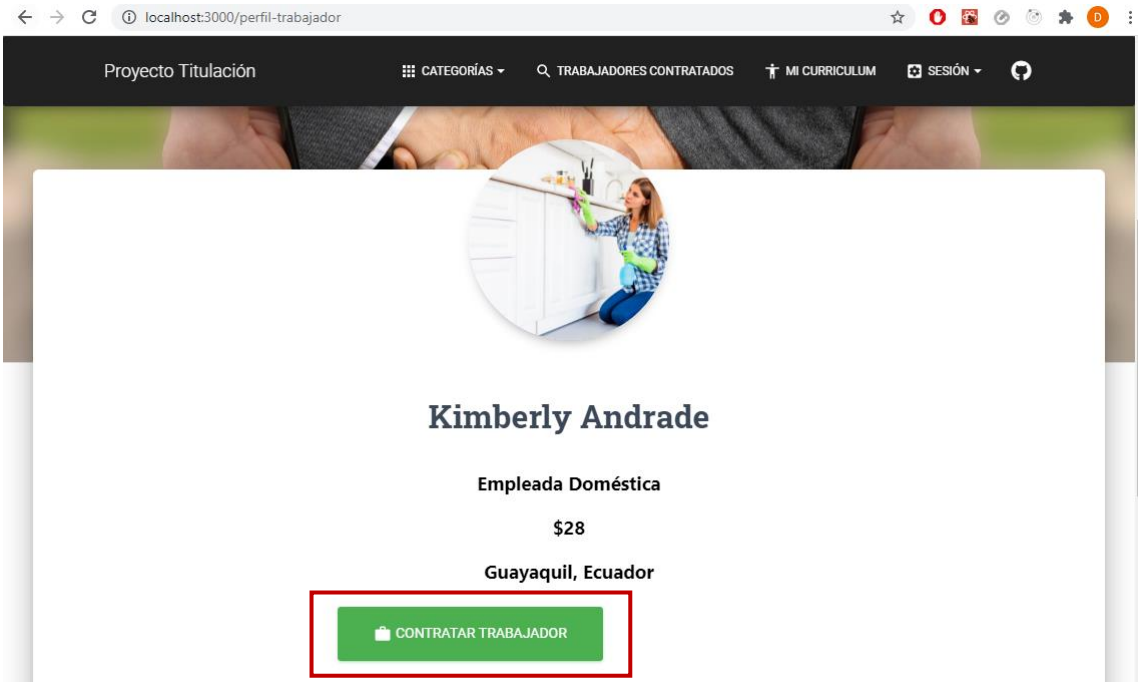
Caso de Prueba	
ID: CAS_05	**Fecha:** 26-Junio-2020
Nombre de historia: Realización de Contratación	
Descripción: Acceder al perfil del trabajador para realizar una contratación y verificar el correcto almacenamiento de cada uno de los datos relevantes de la contratación en Amazon DynamoDB.	
Resultado: 	

Figura 85. Resultado Caso de Prueba Cinco Contratación de Trabajador

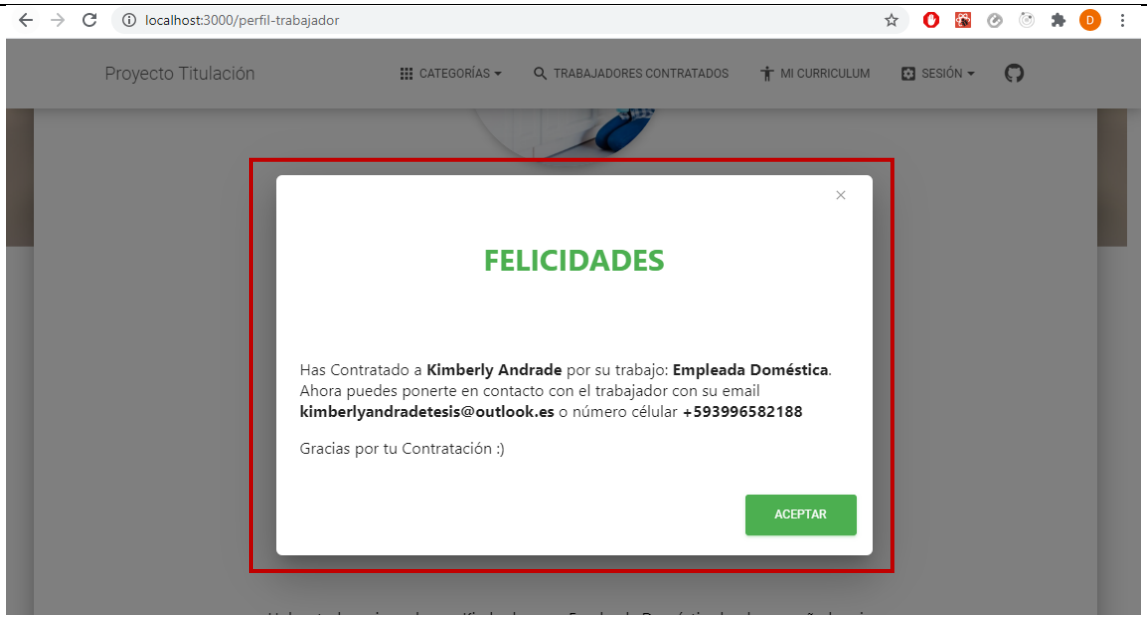


Figura 86. Resultado Caso de Prueba Cinco Despliegue de Modal de Contratación Exitosa

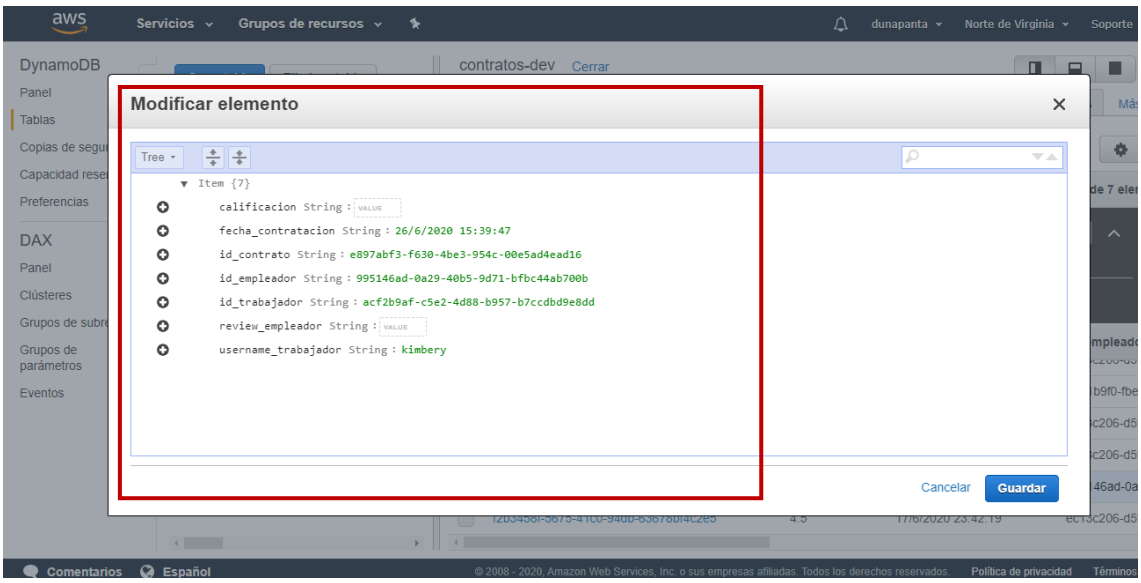


Figura 87. Resultado Caso de Prueba Cinco Verificación de Almacenamiento de Datos de Contratación de DynamoDB

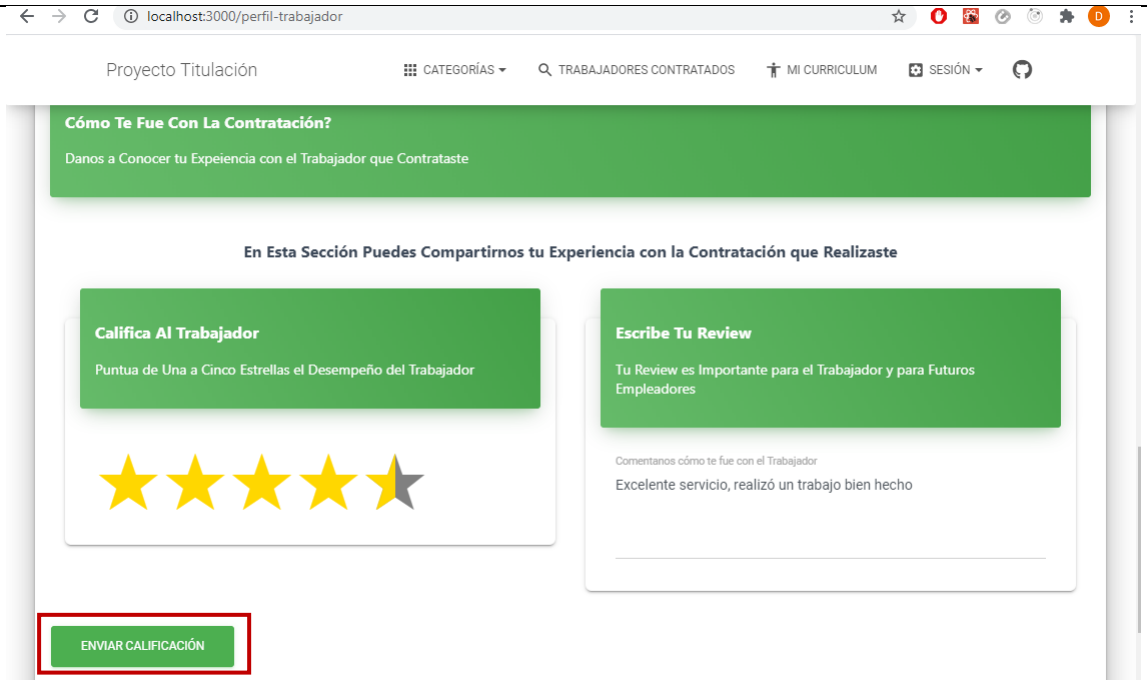


Figura 88. Resultado Caso de Prueba Cinco Envío de Calificación

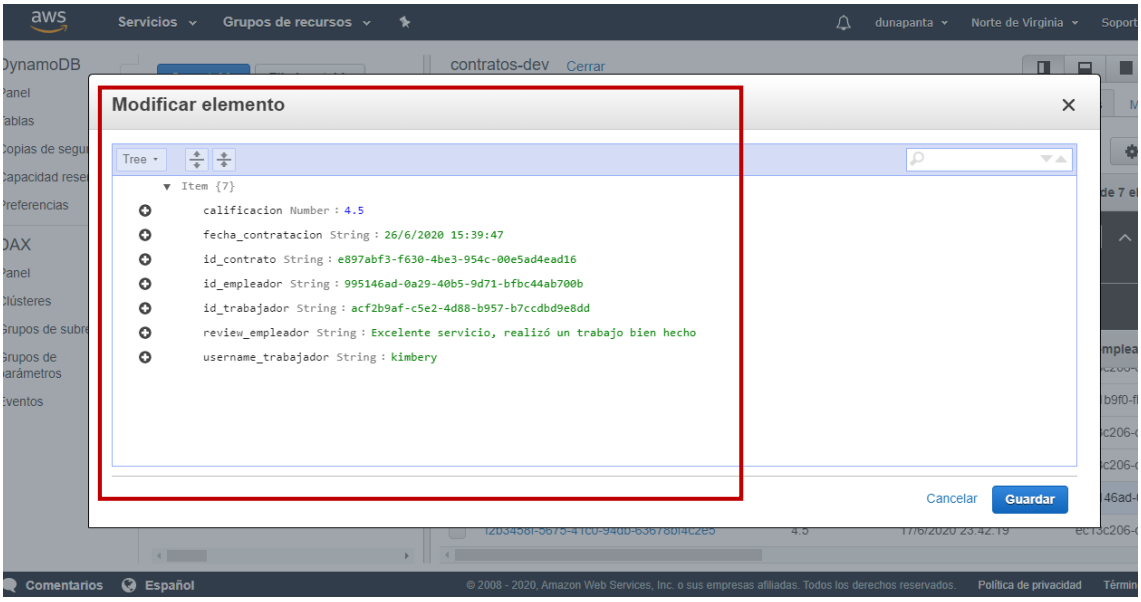


Figura 89. Resultado Caso de Prueba Cinco Verificación de Almacenamiento de Calificación en DynamoDB

5.6 Pruebas del Sexto Sprint

A continuación, se presentan las pruebas realizadas para el sexto Sprint

5.6.1 Prueba Unitaria del Sexto Sprint

Finalmente se presenta la prueba del renderizado correspondiente al componente padre de ListaContrados.

De igual manera se define la prueba por medio de la utilización del método shallow de enzyme y se lo compara con un snapshot del componente ListaContratos para cerciorarse que el renderizado fue correcto.

```
import React from 'react';
import { shallow } from 'enzyme';
import ListaContratos from 'views/ListaContratos/ListaContratos'

describe('Pruebas en Componente de ListaContratos', () => {

    test('Debe mostrar ListaContratos correctamente', () =>{
        const wrapper = shallow(<ListaContratos />)
        expect( wrapper ).toMatchSnapshot();
    })

})
```

Figura 90. Definición de Prueba Unitaria de Renderizado de ListaContratos

A continuación, se presenta el resultado satisfactorio de la prueba obtenida previamente

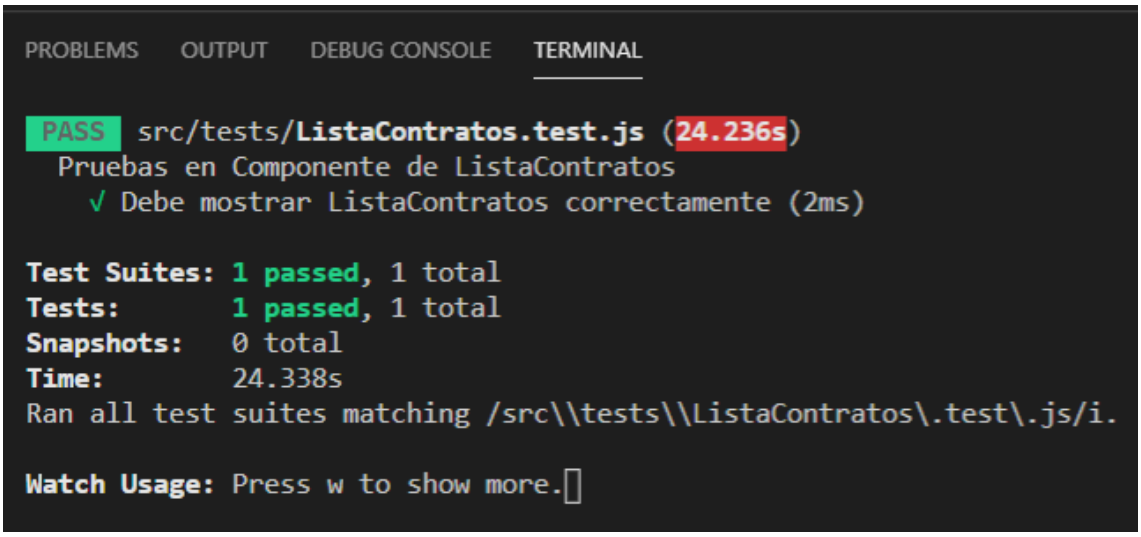


Figura 91. Prueba Unitaria de Renderizado de ListaContratos Satisfactoria

5.6.2 Caso de Prueba del Sexto Sprint

Tabla 27.

Caso de Uso Verificación de Contratación Realizada

Caso de Prueba	
ID: CAS_06	**Fecha:** 26-Junio-2020
Nombre de historia: Verificación de Contratación Realizada	
Descripción: Acceder a la sección de listado de contrataciones y verificar el correcto despliegue de los datos de la contratación realizada.	
Resultado:	

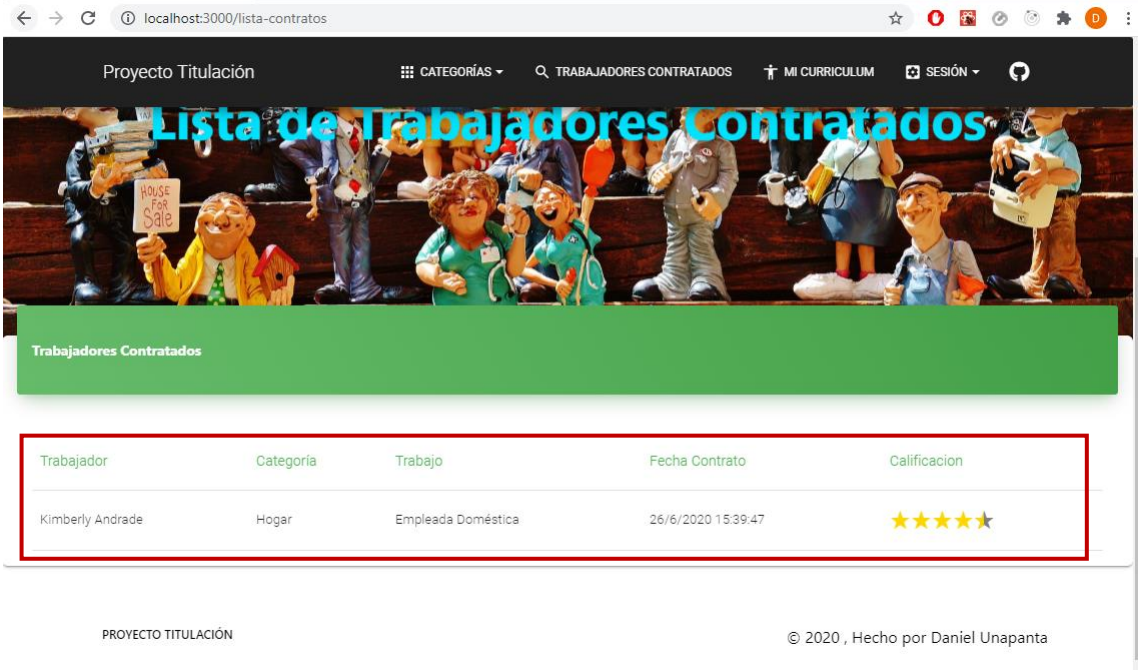


Figura 92. Resultado Caso de Prueba Seis Verificación de Contratación Realizada

6 Conclusiones y Recomendaciones

6.1.1 Conclusiones

En el presente proyecto de titulación se implementó una aplicación web de tipo Single Page Application que hace uso de varios servicios de Amazon Web Services (AWS) para definir las funcionalidades correspondientes a la creación de un curriculum de trabajador para su postulación, despliegue de trabajadores disponibles, contratación de un trabajador y despliegue de contrataciones realizadas.

Se proporcionó una propuesta de aplicación web que hace uso de tecnologías innovadoras de desarrollo web, al establecer el frontend con la utilización de React y Redux y el backend con la definición de la arquitectura Serverless por medio de la interacción de servicios de AWS;

de esta forma que se implementó una aplicación web capaz unificar en una misma plataforma los servicios tanto de postulación para un trabajador como de contratación para un empleador; es así, que se pone a disposición del usuario una alternativa de interés para la generación de empleo.

Se distribuyó el desarrollo correspondiente al frontend de la aplicación web en componentes utilizando la librería React. De esta manera se demostró las ventajas que otorgan los componentes, ya que al ser capaces de administrar su propio estado y su comportamiento independiente; esto facilitó la implementación de interfaces con cierta complejidad en donde es requerido almacenar datos en tiempo de ejecución.

Se definió una arquitectura Serverless para el despliegue del backend utilizando tecnologías de Amazon Web Services como lo son: Amplify para la creación de los servicios necesarios en la nube, Amazon Cognito el servicio encargado de manejar toda la autenticación de los usuarios, API Gateway encargado de la publicación de las API REST necesarias para la obtención de datos de curriculum y contratos, Amazon S3 para almacenar los assets de la aplicación, AWS Lambda para definir las funciones que se ejecuten al momento de realizar alguna petición en la aplicación y Amazon DynamoDB para el almacenamiento de los datos de la aplicación web.

Se aplicó el marco de trabajo ágil Scrum para el desarrollo de la aplicación por medio de iteraciones. De esta forma fue posible evidenciar la importancia y utilidad que tuvo el haber estructurado la planificación y el proceso de desarrollo del proyecto mediante su división en seis sprints y así enfocarse en la presentación de los entregables según el cronograma establecido.

6.1.2 Recomendaciones

Se recomienda la utilización de marcos de trabajo como React, Angular o Vue para la creación de webs de tipo Single Page Application ya que simplifican en gran medida el desarrollo y mantenimiento de dichas aplicaciones.

Se recomienda establecer la arquitectura Serverless que conformará el proyecto antes de comenzar con el desarrollo ya que de esta forma es posible prevenir errores de despliegue en AWS.

Se recomienda que cada componente que conforme la aplicación web este contenido en un directorio específico, de esta manera es posible almacenar los archivos correspondientes al código, estilos y pruebas en un solo paquete y así el proyecto se mantenga organizado.

Se recomienda utilizar alguno de los marcos de trabajo dedicados para la creación de recursos en la nube al momento de la implementación una arquitectura Serverless como lo son Serverless Framework, AWS SAM o Amplify pues simplifican el trabajo de despliegue de los recursos a utilizar.

Se recomienda la utilización de alguna técnica de estimación de historias de usuario como lo es Planning Poker ya que de esta forma se establece la dificultad que representa al equipo el desarrollo del sprint.

Para la creación de aplicaciones web con React se recomienda hacer uso de las últimas características implementadas por la librería, en este caso la utilización de React Hooks, disponible a partir de la versión 16.8 de

React ya que con esta versión es posible definir estados y ciclos de vida de componentes sin la necesidad de definir clases o métodos específicos, de esta forma se simplifica el desarrollo de aplicaciones web.

Se recomienda probar las respuestas obtenidas por las funciones Lambda por medio de la aplicación web o por servidores de terceros para verificar que no existan errores de CORS (Cross Origin Resource Service) ya que probar la función por medio de API Gateway no toma en cuenta este tipo de errores de dominios cruzados.

Se recomienda la utilización de Jest y Enzyme en conjunto para la implementación de pruebas unitarias en los componentes de React ya que simplifican en gran medida la creación y ejecución de pruebas unitarias al poner en disposición del desarrollador una serie de utilitarios y métodos para la definición del código de las pruebas.

7 Referencias

Abramov, D. (2015). *React Redux 7.1 Quick Start*. Recuperado el 18 de Octubre de 2019, de https://react-redux.js.org/introduction/quick-start

Adrien, A. (s.f.). *enzyme-to-json*. Recuperado el 26 de Junio de 2020, de https://github.com/adriantoine/enzyme-to-json#readme

Amazon Web Services. (2020). *¿Qué es Amazon API Gateway?* Recuperado el 16 de Abril de 2020, de https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/welcome.html

Amazon Web Services. (2020). *¿Qué es Amazon Cognito?* Recuperado el 16 de Abril de 2020, de https://docs.aws.amazon.com/es_es/cognito/latest/developerguide/what-is-amazon-cognito.html

Amazon Web Services. (2020). *¿Qué es Amazon S3?* Recuperado el 16 de Abril de 2020, de https://docs.aws.amazon.com/es_es/AmazonS3/latest/dev/Welcome.html

Amazon Web Services. (2020). *¿Qué es AWS Lambda?* Recuperado el 17 de Abril de 2020

Amazon Web Services. (2020). *Amazon Cognito*. Recuperado el 16 de Abril de 2020, de https://aws.amazon.com/es/cognito/

Amazon Web Services. (2020). *AWS Amplify*. Recuperado el 15 de Abril de 2020, de https://aws.amazon.com/es/amplify/

Amazon Web Services. (2020). *Build a Serverless Web Application*. Recuperado el 13 de Abril de 2020, de https://aws.amazon.com/es/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/

Amazon Web Services. (2020). *Sin servidor*. Recuperado el 13 de Abril de 2020, de https://aws.amazon.com/es/serverless/

Amplify Docs. (s.f.). *Sign up, Sign in & Sign out*. Recuperado el 4 de Mayo de 2020, de https://docs.amplify.aws/lib/auth/emailpassword/q/platform/js

Amplify Docs. (s.f.). *Working with API*. Recuperado el 4 de Mayo de 2020, de https://docs.amplify.aws/lib/auth/working-with-api/q/platform/ios

Apiumhub. (1 de Junio de 2017). *A COMPLETE SCRUM SPRINT EXPLANATION*. Recuperado el 22 de Abril de 2020, de https://medium.com/@Apiumhub/a-complete-scrum-sprint-explanation-572200824c17

Babel. (s.f.). *What is Babel?* Recuperado el 17 de Abril de 2020, de https://babeljs.io/docs/en/

Bashir, F. (2018). *What is Serverless Architecture? What are its Pros and Cons?* Recuperado el 13 de Abril de 2020, de https://medium.com/hackernoon/what-is-serverless-architecture-what-are-its-pros-and-cons-cc4b804022e9

Bougère, C. (6 de Agosto de 2019). *AWS Amplify beyond the quickstart*. Recuperado el 16 de Abril de 2020, de https://medium.com/@christophe.bougere/aws-amplify-beyond-the-quickstart-c389f8e44c92

BrandEPS. (2020). *AWS S3 LOGO VECTOR*. Recuperado el 2020 de Abril de 16, de https://www.brandeps.com/logo/A/AWS-S3-01

Bulat, R. (24 de Enero de 2019). *Testing in React with Jest and Enzyme: An Introduction*. Recuperado el 26 de Junio de 2020, de https://medium.com/@rossbulat/testing-in-react-with-jest-and-enzyme-an-introduction-99ce047dfcf8

Chowman, K. (2018). *Hands-On Serverless Computing.* Birmingham: Packt.

Ciel. (17 de Mayo de 2017). *What is Webpack and why should I care? [Part 1][Introduction]*. Recuperado el 17 de Abril de 2020, de https://medium.com/the-self-taught-programmer/what-is-webpack-and-why-should-i-care-part-1-introduction-ca4da7d0d8dc

Copes, F. (13 de Junio de 2018). *A beginner's introduction to Webpack*. Recuperado el 17 de Abril de 2020, de https://medium.com/free-code-camp/a-beginners-introduction-to-webpack-2620415e46b3

Cynix, M. (27 de Agosto de 2019). *What is Amazon DynamoDB?* Recuperado el 17 de Abril de 2020, de https://medium.com/@madhucynixit/what-is-amazon-dynamodb-844b99830773

Dao, N. (2018). *THE SERVERLESS SERIES — What Is Serverless?* Recuperado el 13 de Abril de 2020, de https://blog.neap.co/the-serverless-series-what-is-serverless-d651fbacf3f4

Facebook. (25 de Junio de 2020). *React*. Obtenido de https://github.com/facebook/react

Feijoo, E., & del Pozo, D. (2019). *Encuesta Nacional de Empleo, Desempleo y Subempleo (ENEMDU,) junio 2019*. Recuperado el 13 de Abril de 2020, de https://www.ecuadorencifras.gob.ec/documentos/web-inec/EMPLEO/2019/Junio/Boletin_tecnico_de_empleo_jun19.pdf

Github. (s.f.). *AWS Amplify CLI*. Recuperado el 16 de Abril de 2020, de https://github.com/aws-amplify/amplify-cli

Hernandez, J. (29 de Diciembre de 2019). *What is Amazon S3?* Recuperado el 16 de Abril de 2020, de https://medium.com/@jovanshernandez/what-is-amazon-s3-8362ee26c7c3

INEC. (2018). *Tecnologías de la Información y Comunicación Encuesta Multipropósito - TIC 2018*. Recuperado el 12 de Octubre de 2019, de https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Sociales/TIC/2018/201812_Principales_resultados_TIC_Multiproposito.pdf

JSS. (s.f.). *Introduction*. Recuperado el 17 de Abril de 2020, de https://cssinjs.org/?v=v10.1.1

KindPNG. (2020). *Aws Api Gateway Logo, HD Png Download*. Recuperado el 16 de Abril de 2020, de https://www.kindpng.com/imgv/ibTihRm_aws-api-gateway-logo-hd-png-download/

Klems, M. (2018). *AWS Lambda Quick Start Guide.* Birmingham: Packt.

Lai, C. (25 de Enero de 2019). *Things you should consider before using DynamoDb*. Recuperado el 17 de Abril de 2020, de https://medium.com/@crizantlai/things-you-should-consider-before-using-dynamodb-85b97daf83f5

Material-UI. (s.f.). *MATERIAL-UI*. Recuperado el 17 de Abril de 2020, de https://material-ui.com/

Nawaz, D. (2017). *Serverless: The next level of abstraction.* Recuperado el 13 de Abril de 2020, de https://blog.cloudboost.io/serverless-the-next-level-of-abstraction-30f2003a49e3

Neoteric. (2016). *Single-page application vs. multiple-page application*. Recuperado el 13 de Octubre de 2019, de https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

Python. (15 de Octubre de 2019). *The Python Tutorial*. Obtenido de https://docs.python.org/3/tutorial/index.html

React. (2019). *Glossary of React Terms*. Recuperado el 2019 de Octubre de 13, de https://reactjs.org/docs/glossary.html#single-page-application

React. (2019). *Introducing JSX*. Recuperado el 14 de Octubre de 2019, de https://reactjs.org/docs/introducing-jsx.html

Roberts, M. (2018). *Serverless Architectures*. Recuperado el 17 de Octubre de 2019, de https://martinfowler.com/articles/serverless.html

Sahin, K. (2019). *7 Reasons why you should use React*. Recuperado el 17 de Octubre de 2019, de https://stories.jotform.com/7-reasons-why-you-should-use-react-ad420c634247

Schwaber, K., & Sutherland, J. (Julio de 2013). *La Guía de Scrum*. Recuperado el 22 de Abril de 2020, de https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

Services, A. W. (2020). *AWS Architecture Icons*. Recuperado el 17 de Abril de 2020, de https://aws.amazon.com/es/architecture/icons/

Shevindi, R. (18 de Mayo de 2019). *What is Amazon DynamoDB ?* Recuperado el 17 de Abril de 2020, de https://medium.com/@shevi.rodrigo/what-is-amazon-dynamodb-7df4bb14aa46

Silva, N., González Reina, D., Hereñú, D., & Teruel, P. (2019). *Redux*. Recuperado el 18 de Octubre de 2019, de https://es.redux.js.org/#

Sri, A. (10 de Avril de 2020). *What is AWS Lambda?* Recuperado el 17 de Abril de 2020, de https://medium.com/@jeyanthi.coursejet/what-is-aws-lambda-5bd11154fba6

Techlabs, M. (2017). *What is Serverless Architecture? What are its criticisms and drawbacks?* Recuperado el 13 de Abril de 2020, de https://medium.com/@MarutiTech/what-is-serverless-architecture-what-are-its-criticisms-and-drawbacks-928659f9899a

TechTarget. (2020). *Amazon Cognito*. Recuperado el 16 de Abril de 2020, de https://searchaws.techtarget.com/definition/Amazon-Cognito

Van den Bergh, M. (08 de Septiembre de 2017). *React Redux: Building Modern Web Apps with the ArcGIS JS API*. Recuperado el 17 de Mayo de 2020, de https://www.esri.com/arcgis-blog/products/3d-gis/3d-gis/react-redux-building-modern-web-apps-with-the-arcgis-js-api/

webpack. (s.f.). *Concepts*. Recuperado el 17 de Abril de 2020, de https://webpack.js.org/concepts/

Willem-Jam, A. (17 de Abril de 2020). *What Is Scrum?* Recuperado el 22 de Abril de 2020, de https://medium.com/illumination/what-is-scrum-6096ab5f0ce7

Xalambrí, S. D. (2018). *Introducción a Redux.js.* Recuperado el 20 de Octubre de 2019