Implementación en una aplicación web con ciberseguridad

Ph. D. Barbara Emma Sanchez Rinza[1], stu. Sofronio Arturo Escobar Cortes[2], [1,2] Benemerita Autonomous University of Puebla, *brinza@hotmail.com, s.arturoec@gmail.com*

Resumen **Gran parte de la información que se procesa en las aplicaciones web de hoy en día es privada y altamente sensible. La seguridad, por lo tanto, es un gran problema. Nadie quiere usar una aplicación web si se cree, que su información será revelada a terceros no autorizados. Las aplicaciones web, traen consigo amenazas de seguridad nuevas e importantes, dado que cada aplicación es diferente y puede tener vulnerabilidades únicas**.
Con esto último en mente, da la idea para generar el siguiente trabajo, consiste en primera instancia, crear una aplicación web funcional que ejemplificara un claro motivo del porque es necesario preocuparse por la seguridad en la web. Posteriormente se implementar una solución criptográfica, con un algoritmo asimétrico, para proteger la información sensible que se está intercambiando entre un cliente y el servidor de la aplicación a través de la red.

Keywords—web, ciberseguridad,HTPPS, framework, angular, RSA

I. INTRODUCCION

En los primeros días de internet, la **W*orld Wide Web*** constaba únicamente de sitios web que eran esencialmente, depósitos de información que contenían documentos estáticos. Los navegadores fueros desarrollados como un medio para recuperar y mostrar esos documentos. El flujo de información era unidireccional, este era solo del servidor al navegador. La mayoría de los sitios no autenticaban a los usuarios, ya que no había necesidad de ello, porque cada usuario era tratado de la misma manera y se le presentaba la misma información.
Hoy en día la World Wide Web es prácticamente irreconocible respecto a sus primeros días, ya que la mayoría de los sitios web actuales son de hecho aplicaciones. Son altamente funcionales y se basan en un flujo información bidireccional entre el servidor y el navegador en que la aplicación se esté ejecutando. Entre sus muchas funciones tenemos que admiten registro e inicio de sesión, transacciones financieras, búsquedas y la creación de contenido por parte de los usuarios. El contenido presentado a los usuarios se genera dinámicamente sobre la marcha y, a menudo, se adapta a cada usuario en específico[1].
Con esta evolución de la World Wide Web, surgen un sinfín de novedosas opciones para unirse a la tendencia acuñada por un término denominado WEB 2.0, el cual se refiere al mayor uso de la funcionalidad que permite compartir información y contenido generado por el usuario, y también la adopción de diversas tecnologías que admiten ampliamente esta funcionalidad, incluidas las solicitudes HTPP asíncronas y la integración entre dominios.
Del amplio panorama de tecnologías que tenemos hoy en día para elegir, se decidió para este proyecto tener lo más separado posible la tecnología del cliente (front-end) y la tecnología del servidor (back-end) para crear esa aplicación web inicial. De esta forma se representa el propósito del objetivo y se logró la implementación del aspecto de seguridad con una implementación de un algoritmo de cifrado asimétrico RSA.

A.

LACCEI Proceedings are currently indexed by EBSCO. We are in the process of obtaining additional indexing, which may require additional instructions for the final version of the refereed papers. This section will contain further information as we obtain new indexing for the proceedings.

II. DESARROLLO GENERAL

El primer paso consiste en elegir las tecnologías para trabajar y un algoritmo asimétrico. Para el aspecto enfocado al lado del cliente (front-end) se buscó que la tecnología a elegir debía ser vigente, ampliamente utilizada y enfocada a trabajar con el lenguaje de programación **JavaScript**, dado que este es el único que se ejecuta en los navegadores. Tomando en cuenta lo anterior, la elección fue el framework Angular, auspiciado por la empresa Google Inc[2,3].
En el aspecto enfocado al lado del servidor, tenía que ser una tecnología madura, robusta, segura y escalable, y lo más desapegada posible a tecnologías pensadas para el front-end, por lo tanto se optó por Spring, un framework que es bastante bien posicionado, para la plataforma Java.
Y para la seguridad, se optó por el algoritmo de cifrado asimétrico **RSA**, que nos permite operar con un par de claves, la pública que es compartida con el cliente, y la privada que está siempre en el servidor. Este algoritmo es uno de los utilizados por el protocolo **HTTPS,** con lo cual, fue perfecto para este Proyecto [4,5].

A. Construyendo el front-end con Angular

La aplicación web resultante, no tenía un propósito específico o alguna funcionalidad específica, lo principal a tomar en cuenta, fue la implementación del algoritmo de seguridad RSA en dicha aplicación para proteger información sensible, y resaltar la importancia de la seguridad en la web, con lo cual la aplicación resultante fue una especie de *e-commerce* para una tienda de helados, en donde un determinado cliente, puede hacer lo siguiente[6,7]:

1) Acceder a una página de inicio donde se explica el propósito general de la aplicación.
2) Iniciar sesión con una cuenta ya creada, o crearse una cuenta nueva.
3) Acceder a un apartado D*esign* para diseñar un helado con los ingredientes de su preferencia y posteriormente agregarlo a su carrito.
4) Acceder a un apartado *Latest designs* para elegir diseños anteriores de otros usuarios y agregarlos a su carrito de compras, si así lo desea.
5) Acceder a un apartado *Cart* para finalizar sus pedidos de helados y proceder a pagar, esto lo podemos ver en la figura 1, 2,3,4 y 5.

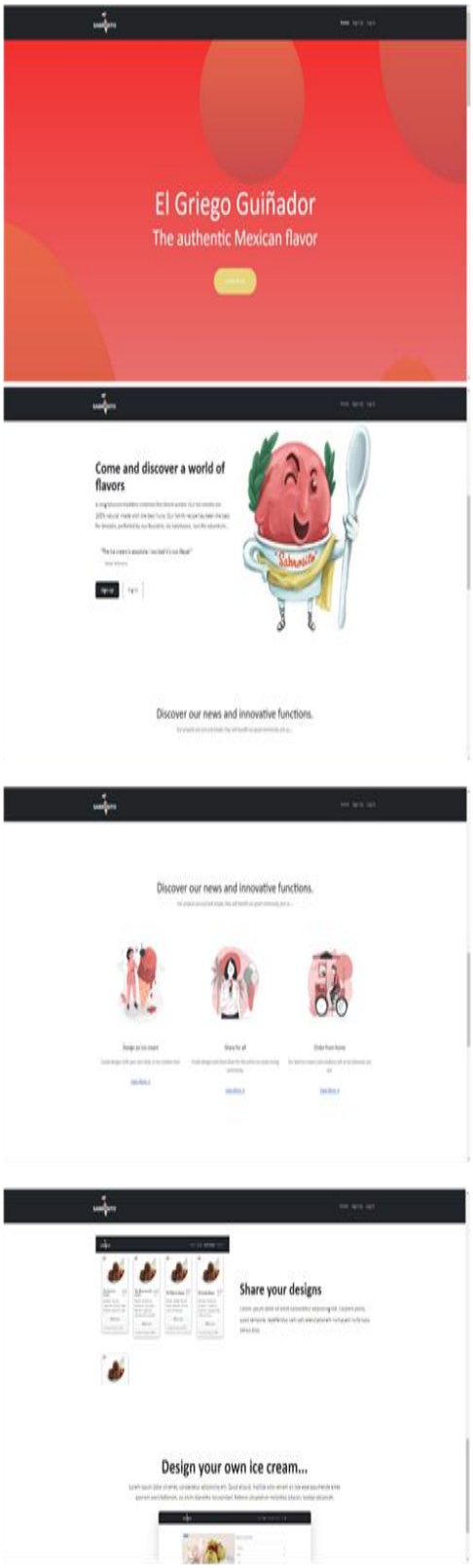

Fig. 1 Interfaz de la página web con el usuario

La figura 1 representa una serie de imágenes, representa el primer contacto que un determinado usuario tiene, al visitar el sitio de la aplicación web.

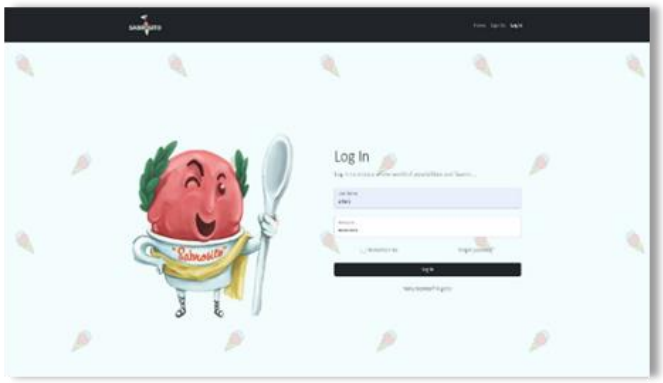

Fig. 2. Usuario accediendo a su cuenta

En la figura 2 representa la vista mostrada a un usuario para acceder a su cuenta previamente creada, para acceder a las distintas funcionalidades de la aplicación web.

Fig. 3 En la figura, representa la vista mostrada a un usuario para crearse una nueva cuenta

Una vez realizado lo anterior, el usuario deberá ingresar sus datos en la vista de la figura para iniciar sesión, ver figura 4.

Fig. 4. La imagen de la izquierda, representa a un usuario ya autenticado

Una vez que se hace la autentificación, mostrando su nombre en con opciones adicionales en la barra de navegación principal de la aplicación web.

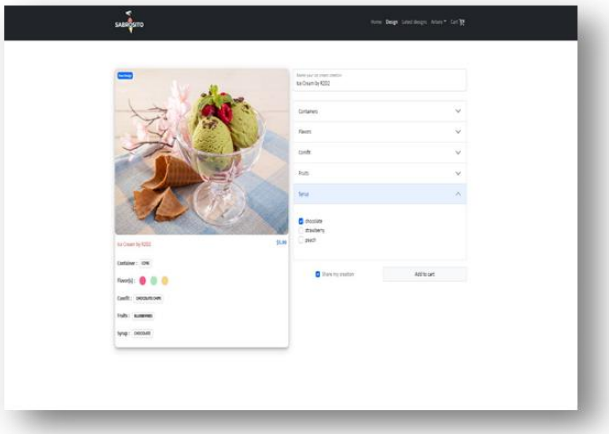

Fig. 5 Representa el apartado Design en la aplicación.

Como podemos ver en la figura 5, un determinado usuario creara sus propios helados personalizados, pudiendo agregarle un nombre a su creación y elegir diferentes opciones de los siguientes rubros:

Contenedor
- Cono
- Vaso
- Canasta
- Copa
- Plato

Sabores
- Vainilla
- Fresa
- Chocolate
- Pistache
- Queso con zarzamoras

Confitería
- Chispas de chocolate
- Cereal
- Malvaviscos
- Chochitos

Frutas
- Cerezas
- Fresas
- Frambuesa
- Arándanos

Jarabe
- Chocolate
- Fresa
- Durazno

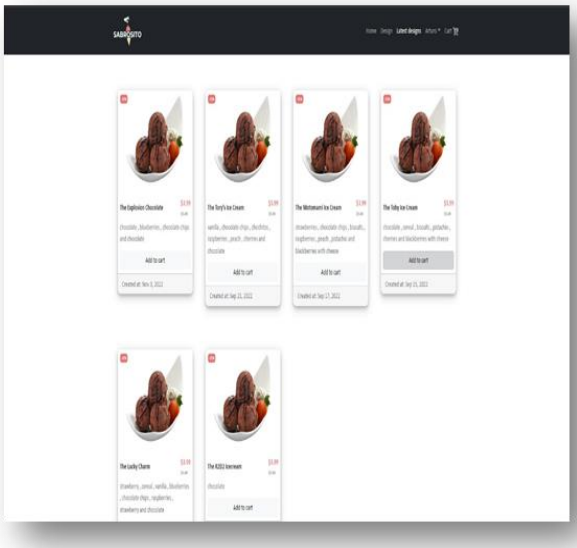

Fig. 6: La imagen de arriba, representa el apartado de Latest designs,

La figura 6 presentaran al usuario los últimos diseños creados y compartidos por otros amantes del helado. El usuario podrá elegir uno o más de estos diseños para agregarlo a su carrito.

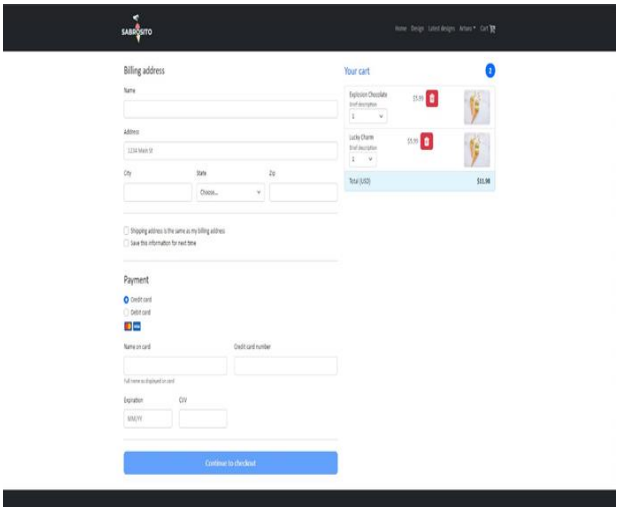

Fig. 7 Muestra el apartado de *Cart*,

La figura 7 el cual es el carrito de compras del usuario, donde ve los helados que desea comprar, y a su vez donde ingresara a su información de pago para completar su compra.
Para el desarrollo de este front-end se utilizó el IDE Visual Studio Code, en conjunto con el framework Angular y sus respectivas dependencias, mismas que se aprecian en la figura 8 [8,9].

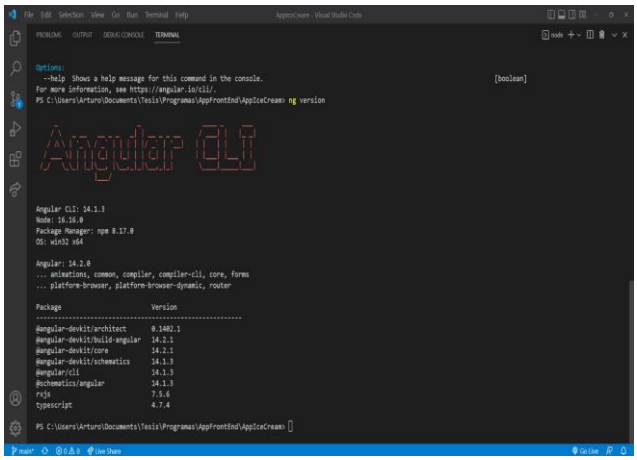

Fig. 8.-codigo de framework angular

El proyecto se gestionó con una estructura generada por la propia herramienta de Angular, (ANGULAR CLI) aplicando todos los conceptos de este framework, como el uso de componentes, servicios, templates etc. Posteriormente se hizo la “build” del proyecto y se puso en marcha en una plataforma de cloud llamada Netlify., ver figura 9 .

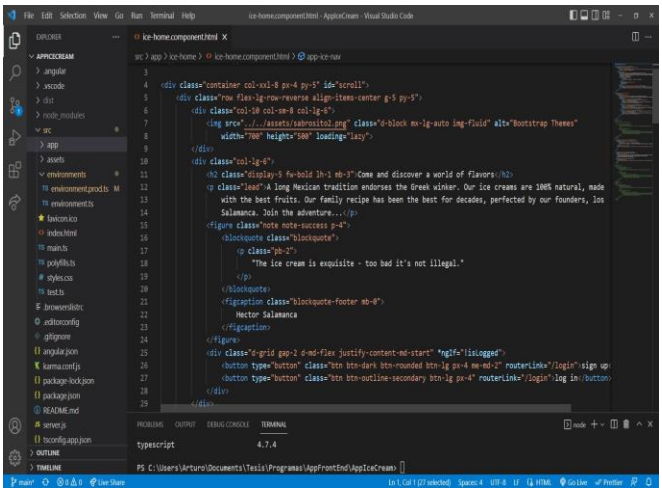

Fig. 9 Parte del código del proyecto

B. Construyendo el back-end con SpringBoot

Una vez sentadas las bases para trabajar con este sistema de *e- commerce*, se puso en manos SpringBoot, el cual ayuda a iniciar un proyecto Spring sin la agonía de la configuración inicial. Valiéndome del IDE Netbeans versión 11.2 se genera la estructura del proyecto y aplica los propios conceptos que el framework ofrece, como lo son los controladores, los repositorios, entidades de dominio de la aplicación, servicios, seguridad etc. ver figura 10 [10].

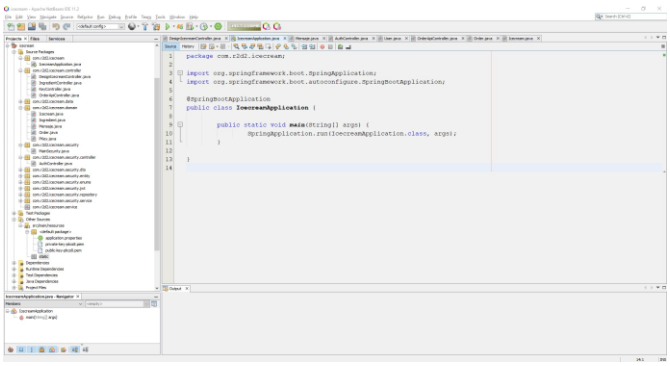

Fig. 10 muestra la estructura del proyecto realizada con SpringBoot y la clase principal que lanza el back-end.

El resultado final, fue una API REST, que responde a peticiones del exterior con respuestas que contienen información en formato JSON. Este formato es recibido por cualquier front-end (Angular en este caso) y es representado en un documento HTML. El siguiente fragmento muestra dicho formato, y representa la información que compone el diseño de un helado creado por un usuario y recuperado desde la base de datos, ver figura 11[11].

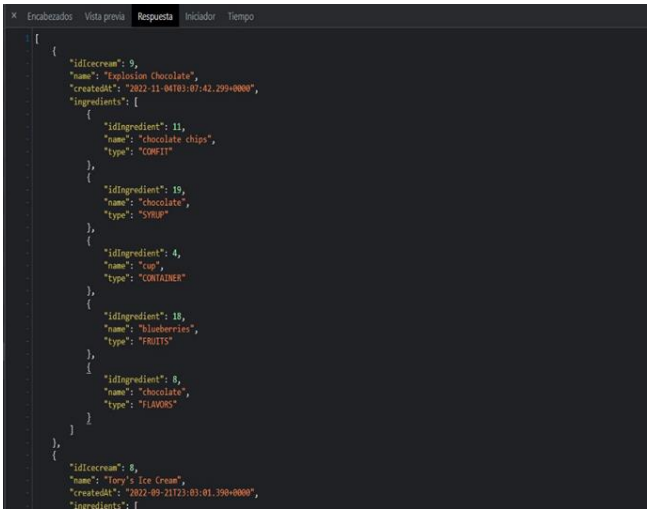

Fig.11 muestra la información de un determinado helado

La figura 11 muestra el ID del helado que tiene en la base de datos, su nombre, la fecha de creación y la serie de ingredientes que lo componen. Con el formato anterior se manejan todos los datos que recibe y envía el cliente realizado en angular. Por parte del servidor, recibe esta información y la transforma en un formato que Java entienda, es decir las convierte en objetos de una clase. Para persistir los datos, Se hizo uso de una herramienta ORM llamada hibernate, que nos permite realizar operaciones en la base de datos de una manera sencilla y rápida. Para la persistencia de la información, se elogió un sistema gestor de base de datos "open source", conocido como PostgreSQL.

Una vez listo todo el andamiaje para el back-end, se procedió a compilar el proyecto y realizar la "build" del mismo, para posteriormente desplegarlo en una plataforma de cloud llamada Heroku, la cual también permite desplegar una base datos PostreSQL, con lo cual en dicha plataforma esta todo lo necesario para el lado del servidor.

III. AGREGANDO SEGURIDAD CON RSA

Para agregar el apartado de la seguridad en el sistema, se debía implementar RSA, existe una biblioteca compatible con angular llamada JSEncrypt, la cual nos permite cifrar y descifrar información con la calve pública y privada respectivamente. Por parte del back-end, Java que es la plataforma sobre la cual trabaja el framework de Spring, ya tiene una implementación en el paquete java.security por lo tanto, también podría cifrar y descifrar con las claves pública y privada.

Para comenzar con la implementación, primero se debe crear el par de claves, la pública y la privada, para ello utilice la herramienta OpenSSL que ofrece una manera rápida y sencilla de hacerlo. Los siguientes comandos crean una clave privada y a partir de esta crean una clave pública:

```
# generate a private key with the correct length
openssl genrsa -out private-key.pem 1024
# generate corresponding public key
openssl rsa -in private-key.pem -pubout -out public-key.pem
```

Una vez generadas las claves, solo fue cuestión de seguir la documentación de ambas tecnologías para asegurar la interoperabilidad e implementar el apartado de seguridad. En general toda la información que se necesite proteger por parte del cliente será cifrada con la clave pública antes de ser enviada. Por su parte el servidor realizara el descifrado con la clave privada para recuperar el texto en claro.

En primera instancia, cuando un usuario se autentica en el servidor, este envía entre otras cosas la clave pública para que sea utilizada en el cliente y realice el cifrado de la información [5].

La figura 12 muestra el contenido de la clave pública enviada por el back-end, con información adicional como un token de autenticación, una serie de roles de autorización y el nombre del usuario autenticado.

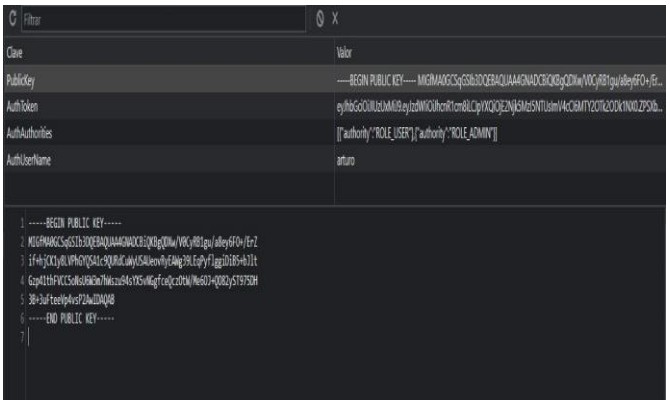

Fig. 12 clave publica

Esta clave pública es la que se emplea para realizar el cifrado las credenciales del usuario al iniciar sesión, como se muestra en la figura 13

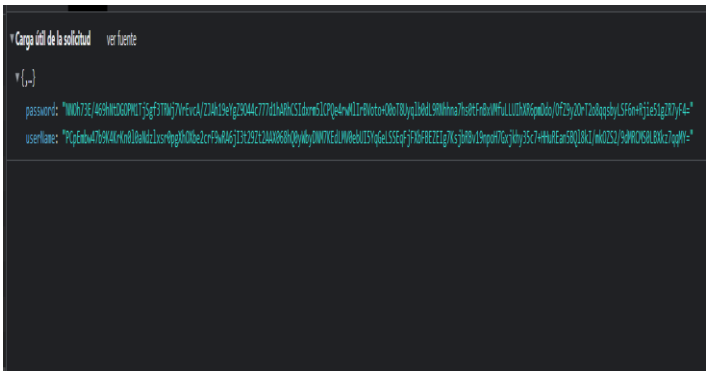

Fig. 13 Cifrado de credenciales del usuario

En el caso de esta aplicación web no toda la información es sumamente sensible, por ejemplo, no parece muy peligroso que una persona no autorizada sepa que me gustan los helados de chocolate con malvaviscos al menos en principio, pero si resulta bastante importante que nadie sepa ni obtenga mis datos bancarios, dirección, código postal etc.

Por esta razón, la información que está contenida en el formulario de pago, es la que nos interesa proteger a toda costa, con lo cual, toda información sensible contenida aquí es cifrada directamente en el cliente, y posteriormente enviada al back-end.

De esta manera, los datos salen de la computadora del usuario viajando por la interred ya cifrados, y si algún curioso quisiera ver esta información, seria ininteligible y bastante difícil de descifrar, ya que estamos respaldados por un algoritmo bastante fuerte y ampliamente utilizado en el protocolo HTTPS, ver figura 14.

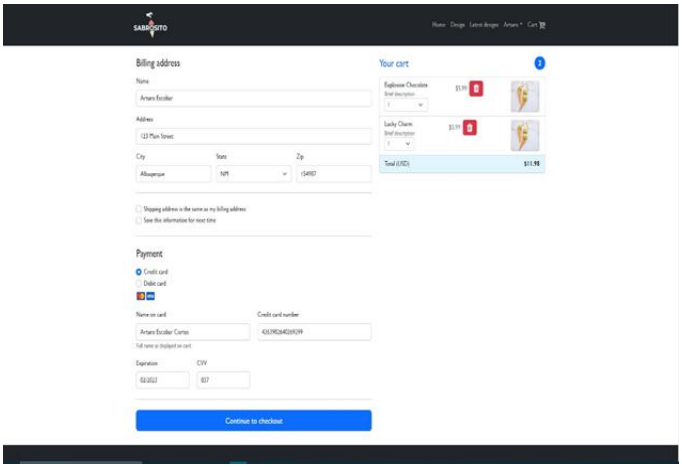

Fig. 14 pantalla donde se ve al usuario meter su información personal.

La figura 15 representa la información capturada y cifrada en el formulario de pago.

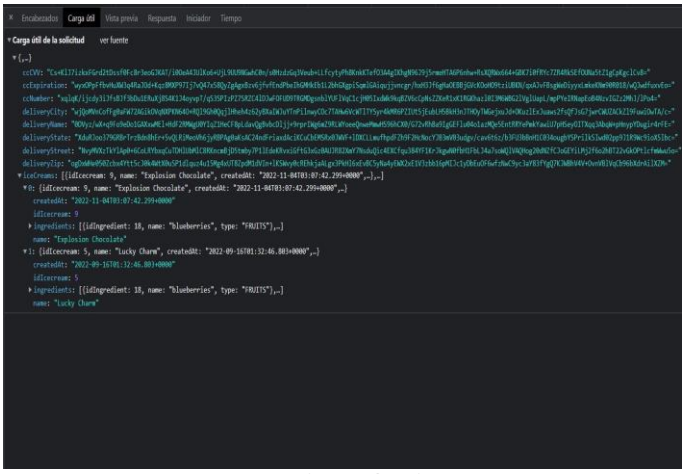

Fig. 15 información cifrada

IV CONCLUSIONES

El trabajo se concluyó satisfactoriamente, dando seguridad a una página comercial, que realiza a diario transacciones económicas y por ende los usuarios tienen que proteger sus datos personales como nombre, número de tarjeta, banco, si la tarjeta es de crédito o de débito, etc. y para el cual se tuvo que echar mano de la siguiente herramienta

1. Se utilizó tecnología detrás del framework Angular con el apartado web que utiliza la plataforma de Java y todo el entorno del framework Spring.
2. Se utiliza el algoritmo RSA que conlleva al manejo de la teoría de números
3. Funcionan los certificados que utiliza el protocolo HTTPS

4. SSL/TLS es una herramienta muy útil para agregar seguridad a la web, No obstante, en algunas ocasiones estos certificados podrían tener un costo y no todos los sitios pueden optar por usar un certificado y por ende no usar el protocolo seguro HTTPS.

REFERENCES

[1] Charles P. Pfleeger, Shari Lawrence Pfleeger y Jonathan Marguiles. (2015). Security in Computing Fifth Edition. Prentice Hall.

[2] Bruce Schneier. (2015). Aplied Cryptography, Protocols, Algorithms and Source Code in C. WILEY.

[3] Dafydd Stuttard y Marcus Pinto. (2011). The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, Second Edition. WILEY.

[4] William Stalligns. (2017). Cryptography and Network Security Principles And Practice Seventh Edition Global Edition. Pearson Education Limited.

[5] Jamsa K. (2002). Hacker proof: the ultimate guide to network security, 2nd edn. Delmar Cengage Learning.

[6] Lincoln D. Stein. (1998). Web security: a step-by-step reference guide. Addison-Wesley.

[7] Onyszko T. July 19, 2002 Secure socket layer: authentication, access control and encryption. http://www.windowsecurity.com/articles-tutorials/authentication_and_encryption/ Secure_Socket_Layer.html

[8] Shastack Adam. May 1995. An overview of S-HTTP. http://www.homeport.org/~adam/shttp.html

[9] Migga Kizza Joseph. (2019). Guide to Computer Network Security. Springer.

[10] Katz Jonathan y Lindell Yehuda. (2015). Chapman and Hall Introduction to Modern Cryptography. CRC Press.

[11] Barbara Emma Sanchez Rinza,(2021), Electro, vol 43. pp 120-124, ISSN 1405-2172