CIS1810CP02

EasyTicket

José Rafael Domínguez Nolasco
Juan Pablo Rodríguez Navarro
María Camila Vanegas Cubillos
Sebastián Bobadilla Zubieta

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
2018

CIS1810CP02

EasyTicket

Autores:

José Rafael Domínguez Nolasco
Juan Pablo Rodríguez Navarro
María Camila Vanegas Cubillos
Sebastián Bobadilla Zubieta

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA CUMPLIR UNO DE LOS REQUISITOS Y OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

Director

MSc Julio Ernesto Carreño Vargas

Jurados del Trabajo de Grado

Sitio Web del Trabajo de Grado

http://pegasus.javeriana.edu.co/~CIS1810CP02

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
NOVIEMBRE, 2018

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
SYSTEMS ENGINEERING PROGRAM

Rector Pontificia Universidad Javeriana

Jorge Humberto Peláez Piedrahita, S.J.

Decano Facultad de Ingeniería

Ing. Lope Hugo Barrero Solano

Director de la Carrera de Ingeniería de Sistemas

Ing. Mariela Josefina Curiel Huérfano

Director Departamento de Ingeniería de Sistemas

Ing. Efraín Ortíz Pabón

Artículo 23 de la Resolución No. 1 de junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

Agradecemos a la Universidad por brindarnos un espacio de aprendizaje, para crecer como personas y profesionales. Agradecemos a nuestro director por el compromiso y constante apoyo a lo largo del desarrollo del trabajo de grado.

CONTENIDO

ABSTRACT

EasyTicket proposes a solution to the current problems related to ticket sales, together with a scalable software architecture that uses current technologies such as serverless computing and mobile applications. Bar codes are used such as PDF417, present in Colombian identity documents, and QR codes to identify the tickets and maintain a pleasant user experience free of problems related to physical establishments.

RESUMEN

EasyTicket propone una solución a los problemas actuales relacionados con la venta de boletería junto a una arquitectura de software escalable que incluya tecnologías actuales como computación sin servidor, aplicaciones móviles y aplicaciones de una sola página. Se utilizan códigos de barras, como PDF417, presente en documentos de identidad colombianos y códigos QR para identificar los boletos y mantener una experiencia agradable para el usuario.

I- INTRODUCCIÓN

La actual venta y distribución de boletería en Colombia presenta cierta dificultad para los clientes, puesto que los proveedores de dicho servicio no brindan las suficientes alternativas al cliente para la obtención de boletas de manera ágil. Problemas como la reventa de tiquetes, falta de garantías una vez adquirido el tiquete y filas a la hora de obtener las boletas son pruebas de la inconformidad de los asistentes con las actuales plataformas de venta.

Con el objetivo de mejorar la experiencia del usuario al adquirir una boleta para un evento, surge la necesidad de crear un sistema de venta de boletería y control de acceso totalmente digital. En el cual no son necesarias las boletas físicas, sino que con ayuda de códigos de barras es posible identificar un tiquete adquirido por un usuario. Se plantea el uso del código PDF417, presente en los documentos de identidad colombianos, así como el código QR, para ser escaneados al momento de ingresar a un evento.

En el presente documento se propone una solución a la problemática actual relacionada con la venta de boletería, junto a una arquitectura de software escalable que incluya tecnologías actuales como computación sin servidor, aplicaciones móviles y aplicaciones de una sola página. El documento se compone de capítulos que contextualizan la problemática actual, presentan algunas de las plataformas de venta en Colombia y también describen el análisis, diseño y desarrollo de la solución propuesta.

II- DESCRIPCIÓN GENERAL

1. Oportunidad, Problemática y Antecedentes

1.1. Contexto del problema

La actual venta y distribución de boletería para eventos ha generado cierta dificultad para el cliente, puesto que los proveedores de dicho servicio no le brindan alternativas al cliente para que pueda conseguir boletas de manera ágil y oportuna [1].

Se ha identificado que no solamente al cliente le afecta la pérdida de tiempo sino la poca disponibilidad de boletas en los puntos de venta y horarios de atención reducidos. En consecuencia, se ha llegado al punto que el cliente no asista al evento, debido a que el servicio no cuenta con las garantías necesarias [2].

Igualmente, en eventos de asistencia masiva el agotamiento de boletas es rápido, debido a que son adquiridas por terceros que buscan un lucro mayor con la reventa de las mismas [3].

Con el propósito de simplificar la venta de boletas, se hace indispensable implementar procesos que brinden al usuario una experiencia sin filas al momento de la compra, y sin precios elevados a causa de los revendedores. La implementación de nuevas tecnologías conlleva a que estos procesos sean más controlados y ofrezcan una solución orientada al beneficio de los clientes.

1.2. Formulación del problema

Algunos de los problemas de la venta de boletería en Colombia son:

- Revendedores:
    - Personas que agotan la boletería oficial y se aprovechan de esta situación para vender las entradas a un precio mucho más elevado.
- Tiquetes físicos:
    - Documentos emitidos por la empresa prestadora del servicio, y recibidos por el cliente.
    - Por su naturaleza tangible, el cliente es el único responsable de su cuidado.

  - Su transferencia no es controlada después de su compra.

- Puntos de venta:
  - Filas al momento de comprar las boletas.
  - Horarios poco flexibles.

Teniendo en cuenta los problemas enunciados ¿cómo emplear tecnologías existentes para ofrecerle al cliente las garantías necesarias para que la compra de boletería no sea un motivo de desagrado o frustración al momento de asistir a un evento?

1.3. Solución propuesta

La solución desarrollada consistió en una plataforma web que permite la venta de boletería en línea, donde es posible que los usuarios adquieran boletas para eventos de pequeña escala, es decir, alrededor de 100 personas. Además, se desarrolló una aplicación móvil enfocada en el control de acceso, la cual permite gestionar el ingreso de los usuarios al evento, gracias al análisis del código de barras PDF417 incluido en los documentos de identidad [4] y códigos QR.

Por medio de las cámaras de los dispositivos móviles, fue posible obtener la información de los códigos de barras, e identificar a los usuarios con el objetivo de permitir su acceso a los eventos.

1.4. Justificación de la solución

En los últimos años, la tecnología de los códigos de barras, específicamente los códigos PDF417 y lo códigos QR, han tenido un progreso importante y han participado activamente en el desarrollo de diferentes áreas a nivel mundial [5] [6]. En el año 2000, el gobierno de Colombia inició la producción de una nueva cédula de ciudadanía, la cual cuenta con un código de barras en formato PDF417 que contiene la información básica del ciudadano [7].

La tecnología mencionada anteriormente, puede ser aprovechada para almacenar otro tipo de información asociada al ciudadano correspondiente sin generar incomodidades o documentos adicionales. Por tal motivo, se utilizó esta tecnología para generar una mejor experiencia de usuario en la venta de boletería y control de acceso a eventos no masivos, teniendo en cuenta la disponibilidad de los documentos de identidad [8].

Teniendo en cuenta que la mayoría de los ciudadanos colombianos mayores de edad portan su cédula de ciudadanía [9], se empleó dicho recurso para crear un sistema modular y escalable, compuesto por una plataforma web y una aplicación móvil. Permite la venta de boletas para eventos no masivos, cuyo control de acceso se realiza por medio de la lectura del código de barras respectivo.

Aunque se espera que la mayoría de los usuarios utilicen su documento de identidad para la interacción con el sistema, se planteó la utilización del código QR como alternativa, en caso de no contar con dicho documento.

2. Descripción del proyecto

2.1. Objetivo general

Desarrollar un prototipo arquitectural para el sistema de venta de boletería y control de acceso, basado en la lectura de códigos de barras, para eventos de pequeña escala.

2.2. Objetivos específicos

- Realizar el diseño de los componentes del sistema.
- Desarrollar un prototipo funcional para el sistema de venta de boletería y control de acceso.
- Validar el diseño y el prototipo mediante una prueba piloto.

2.3. Entregables, estándares y justificación

Entregable	Estándares asociados	Justificación
Plan de proyecto	ISO/IEC/IEEE 16326	Especifica los planes que componen el proyecto, al igual que el cronograma de este
Especificación de requerimientos	ISO/IEC/IEEE 29418	Detalla las funcionalidades que estarán presentes en el proyecto, al igual que los procesos necesarios para su administración
Descripción de diseño	IEEE 1016	Contiene el diseño de la solución, tanto la arquitectura como el diseño detallado de los componentes

Documento de pruebas	ISO/IEC/IEEE 29119	Presenta el diseño, ejecución y resultados de las pruebas funcionales y de aceptación realizadas
Documentación de la arquitectura	Software Arquitecture Document (SAD)	Presenta las principales vistas arquitecturales del sistema para describir su estructura y funcionamiento
Manual de usuario	No aplica	Contiene información de las funcionalidades del sistema para dar asistencia a los usuarios del sistema
Otros estándares		
Código de barras PDF417	ISO/IEC 15438	Definición las especificaciones para la generación y lectura de un código PDF417
Código QR	ISO/IEC 18004	Especificación de la simbología usada en los códigos QR

Tabla 1 Entregables y estándares

III- CONTEXTO DEL PROYECTO

1. Conceptos relevantes

CONTROL DE ACCESO

Se entiende como el proceso por el que un recurso, físico o virtual, es regulado de acuerdo a políticas de seguridad, con el fin de permitir su uso únicamente por parte de entidades autorizadas [10].

Para controlar el ingreso a un espacio físico, son usados sistemas electrónicos encargados de automatizar el proceso de verificación mediante la lectura de algún medio, ya sea una clave, tarjeta o características biométricas de un usuario. Una vez se realiza la lectura, se valida la identificación presentada y se habilita un recurso específico (puerta, torniquete) para permitir el ingreso.

Los sistemas de control de acceso a recursos físicos se clasifican en dos tipos:

- Sistemas de Control de Acceso Autónomos: permiten controlar uno o más recursos, sin estar conectados a un sistema central. Por lo tanto, no guardan registro de eventos. Aunque esta es la principal limitante, algunos controles de acceso autónomos tampoco pueden limitar el acceso por horarios o por grupos de puertas, esto depende de la robustez del sistema.

- Sistemas de Control de Acceso en Red: se integran a través de un dispositivo local o remoto, donde se hace uso de un software de control que permite llevar un registro de todas las operaciones realizadas sobre el sistema con fecha, horario, autorización, etc. Van desde aplicaciones sencillas, hasta sistemas muy complejos y sofisticados según se requiera [11].

CÓDIGO DE BARRAS

Medio de codificación de información usado para describir características del elemento portador del código. Se compone de patrones geométricos que pueden ser leídos mediante escáneres ópticos especializados o software para dispositivos con cámara.

Es uno de los métodos más usados para la identificación y captura de datos automática (AIDC), en los que no es necesaria la intervención humana para el ingreso de información en un Sistema [12]. Por esta razón, los códigos barras son usados ampliamente para identificación de productos y transferencia de información. Además, sus

costos de implementación son menores a los de tecnologías similares como la identificación por radiofrecuencia (RFID).

ESCALABILIDAD:

Propiedad deseable en un sistema, red o proceso que indica su habilidad para poder hacerse más grande sin perder calidad en sus servicios [13]. La escalabilidad debe formar parte del proceso de diseño ya que no es una característica separada que pueda agregada al finalizar la implementación de un sistema. Al igual que con otras funciones de aplicación, las decisiones que se tomen durante las primeras fases de diseño y codificación determinarán en gran medida la escalabilidad de la aplicación [13].

ARQUITECTURA DE MICROSERVICIOS:

Estilo arquitectural que busca dividir una aplicación en un conjunto de servicios independientes, desarrollados alrededor de las principales funcionalidades de negocio y con capacidad de ser desplegados de manera individual, o incluso, ser implementados en tecnologías diferentes [14].

Surge como respuesta a las principales falencias de las aplicaciones monolíticas, en las que existe un único proceso responsable de manejar las solicitudes realizadas por clientes, ejecutar lógica de negocio, interactuar con base de datos y administrar las vistas que son mostradas a los usuarios. Con el incremento de tecnologías en la nube, el despliegue de estas aplicaciones resulta en complicaciones ya que la modificación de alguno de sus componentes implica el despliegue del sistema en su totalidad [14].

ARQUITECTURA SERVERLESS:

Estilo que permite el desarrollo de aplicaciones y servicios sin la necesidad de administrar hardware ni contar con un servidor web. La administración y provisión de hardware, al igual que el escalamiento del software, son provistos por terceros [15].

En esta arquitectura, también conocida como Función como Servicio (FaaS), el sistema se compone de funciones autónomas y el proveedor de la infraestructura se encarga del escalamiento de cada una de estas según la demanda. Con esta aproximación se busca que el desarrollo se enfoque en las funciones del sistema y no en la administración de recursos como servidores, sistemas operativos y servidores web [16].

2. Análisis del contexto

A continuación, se presentan tres diferentes plataformas que actualmente en Colombia se dedican a la venta de boletería bajo un sistema similar, TuBoleta, eTicket y TicketShop. Con el fin de analizar estos sistemas, se especificará su funcionamiento, el problema que resuelve y qué carencias pueden presentar[17] [18] [19].

Los sistemas ofrecen el servicio de distribución y comercialización de boletas de espectáculos y eventos, es decir, teatro, deportes, conciertos, entre otros, mediante diversos canales de venta como aplicaciones para teléfonos móviles, tablets, venta presencial y telefónica. Se da lugar a una adecuada y rápida comercialización de la boletería para los promotores.

En general, estos sistemas resuelven la necesidad de ofrecer boletería de seguridad para la venta de eventos, gracias a los procesos en línea y otros medios de acceso. No obstante, mientras se ofrecen estos canales de venta, estas plataformas requieren que el usuario deba desplazarse a reclamar su boleta, además de que no existe un control sobre cuántas boletas puede comprar un único usuario, lo que podría dar lugar a la reventa malintencionada de las boletas.

Con EasyTicket se propone un canal seguro de venta de boletas digitales, evitando que el usuario final tenga que desplazarse a recoger la boleta, además de tener control sobre la venta de las boletas al gestionar los dueños de cada una de estas. Asimismo, al ser un sistema totalmente digital, es posible generar reportes de análisis de datos de interés para los organizadores de los eventos. En la Tabla 2 se presenta la comparación de los actuales sistemas de boletería.

	TuBoleta	**eTicket**	**Ticket Shop**
Método de venta	En línea o en local	En línea	En línea
Formato de boletas	Físicas, proporcionadas únicamente por la empresa	Imprimibles por el usuario, proporcionadas por la empresa	Físicas, proporcionadas únicamente por la empresa, entrega a domicilio
Uso de códigos de barras	Sí	Sí	Sí
Documentos de identidad como boleta	No	No	No

Tabla 2 Comparación de sistemas actuales

IV- ANÁLISIS DEL PROBLEMA

Teniendo en cuenta el contexto actual de la problemática y la solución planteada anteriormente, en este capítulo se pretende presentar de manera general el análisis arquitectural de la solución, así como describir las principales funcionalidades con las que cuenta el proyecto.

1. Contexto del sistema

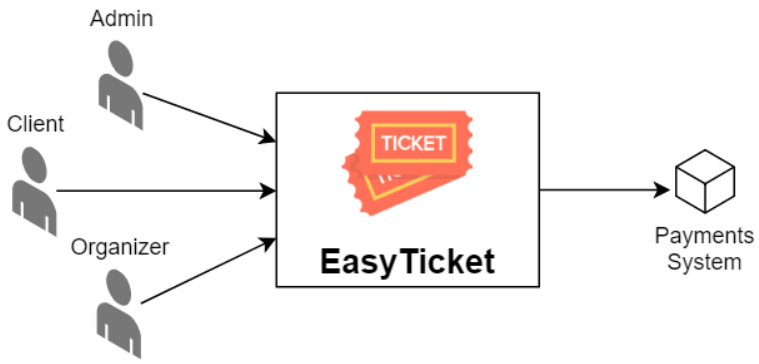

Figura 1 Contexto del sistema

Como se muestra en la Figura 1, los siguientes actores interactúan con el sistema:

- Admin: usuario administrador del sistema, encargado de modificar parámetros del sistema.
- Client: cliente registrado en la plataforma, puede comprar los tiquetes para los eventos existentes.
- Organizer: usuario organizador de eventos, encargado de crear y administrar eventos, así como de controlar el acceso a los mismos.
- Usuario anónimo: usuario no registrado en el sistema, puede ver información básica de los eventos publicados.

Para el funcionamiento se hace necesaria la comunicación con sistemas externos que incluyen:

- Payments system: plataforma de pagos en línea para realizar la compra de tiquetes.

Se cuenta con sistemas de almacenamiento, mensajería asíncrona y envío de correos provistos en la nube. No se incluyen en la figura ya que su configuración hace parte de la implementación del sistema.

2. Contexto arquitectural

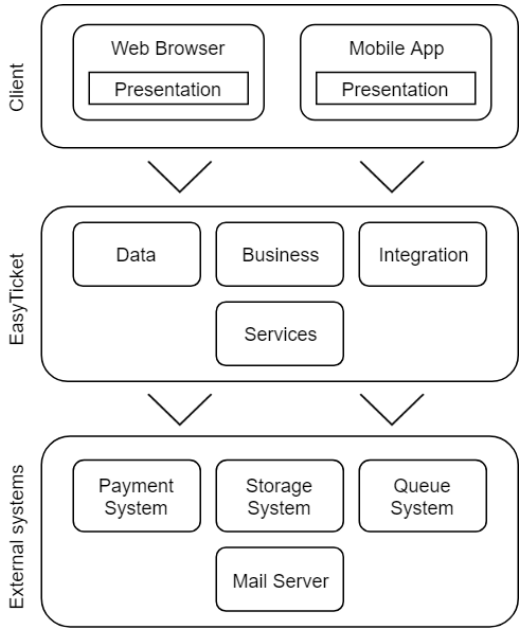

Figura 2 Vista general de la arquitectura

En la Figura 2 se agrupan de manera general los elementos que componen el sistema, descritos a continuación:

CLIENT: contiene los componentes con los que interactúan los usuarios a través de una interfaz gráfica, de igual manera consumen la lógica de negocio.

- Web Browser: los usuarios del sistema (*Admin*, *Client* y *Organizer*) acceden a las funcionalidades de eventos y venta de boletería a través de la página web.

- Mobile App: los usuarios *Organizer* acceden a las funcionalidades de control de acceso a eventos y validación de boletas a través de la aplicación móvil del sistema.

EASYTICKET: sistema en el cual se exponen las funcionalidades de negocio mediante servicios.

- Business: capa de negocio, define la lógica necesaria para el funcionamiento esencial del sistema.
- Integration: capa que integra EasyTicket con otros sistemas externos.
- Data: capa que contiene los modelos y entidades.
- Services: capa que expone las funcionalidades desarrolladas en la capa de negocio.

EXTERNAL SYSTEMS: sistemas externos a EasyTicket que prestan servicios a este. Incluye el sistema de pagos para la compra de tiquetes, servidor de correos para el envío de información a los usuarios. De igual manera se cuenta con el sistema de mensajería para realizar tareas de manera asíncrona.

3. Requerimientos

Dado que la solución surge a una problemática percibida por el grupo de trabajo, y no a una necesidad de un cliente, el proceso de obtención de requerimientos responde a funcionalidades deseadas en la solución descrita en los capítulos anteriores.

A continuación, se detallan los requerimientos principales del sistema, es decir, aquellos que son necesarios para su funcionamiento básico. Se utiliza como referencia la nomenclatura utilizada en el documento de Especificación de Requisitos de Software (SRS) del sistema, donde se detallan a fondo el resto de los requerimientos.

REQUERIMIENTOS FUNCIONALES:

ID	RF11	**Nombre**	Comprar tiquete
Precondición	La sesión debe estar iniciada y debe existir el evento en cuestión		
Descripción	El usuario inicia el proceso de compra de un tiquete asociado a un evento. En caso de no terminar la transacción, se informa al usuario		
Postcondición	El tiquete se asocia con el documento de identidad del usuario y se le informa el resultado de la compra		

Tabla 3 RF11 Comprar tiquete

La compra de tiquetes es una de las actividades básicas del sistema. Permite la conexión entre los usuarios Asistentes a eventos con los Organizadores a eventos, a causa de que los asistentes pueden inscribirse a los eventos propuestos por los organizadores.

ID	RF17	**Nombre**	Generar código QR
Precondición	El usuario debe hacer realizado la compra de una boleta y solicitado la generación del código		
Descripción	El sistema crea el código asociado a un tiquete		
Postcondición	El sistema envía al usuario, por correo electrónico, el código QR asociado al tiquete		

Tabla 4 RF17 Generar código QR

Al generar una boleta se genera un código QR, el cual posee la información necesaria para que pueda ser identificada como única, mediante la aplicación móvil de los Organizadores de eventos.

ID	RF18	**Nombre**	Validar tiquete
Precondición	El cliente debe haber comprado un tiquete		
Descripción	Se escanea el código de barras presentado por el usuario (PDF417 o QR) para verificar la validez del tiquete		
Postcondición	Se actualiza el estado del tiquete par indicar que ya fue presentado		

Tabla 5 RF18 Validar tiquete

La validación se realiza cuando la aplicación móvil de los Organizadores de eventos lee y procesa el respectivo código de barras asociado con la boleta, luego, este se verifica en el servidor con el fin de determinar si la boleta es válida, o no.

ID	RF06	**Nombre**	Crear evento
Precondición	El usuario debe haber iniciado sesión		
Descripción	El usuario proporciona los datos necesarios para la creación del evento		
Postcondición	Un nuevo evento es agregado al sistema		

Tabla 6 RF06 Crear evento

La creación del evento es llevada a cabo por el Organizador, se definen todos los datos necesarios para su inscripción en el sistema y dar lugar la compra de boletería por parte de los clientes.

REQUERIMIENTOS NO FUNCIONALES:

ID	RNF01	**Atributo de calidad**	Rendimiento
Descripción	El sistema debe responder en un tiempo promedio de tres segundos en todas sus operaciones		

Tabla 7 RNF01 Tiempo de respuesta

Algunas fuentes afirman que un tiempo de espera mayor a 4 segundos, puede afectar negativamente la experiencia de los usuarios [20].

ID	RNF02	**Atributo de calidad**	Seguridad
Descripción	Las contraseñas de los usuarios deben estar almacenadas bajo una función de hash		

Tabla 8 RNF02 Contraseñas de usuarios bajo función hash

Con el fin de mantener seguridad en los datos de los usuarios, se espera utilizar una función de hash al momento de almacenar las contraseñas, para evitar que l a información de los usuarios se vea comprometida [21].

ID	RNF03	**Atributo de calidad**	Portabilidad
Descripción	La plataforma web debe funcionar en las últimas versiones de Google Chrome y Firefox		

Tabla 9 RNF03 Compatibilidad de navegadores

Según diferentes fuentes, Chrome y Firefox son los navegadores multiplataforma más utilizados de la web [22]. Por lo tanto, se espera que el sistema funcione correctamente en estos exploradores.

ID	RNF06	**Atributo de calidad**	Seguridad
Descripción	El sistema debe ser capaz de determinar la autenticidad del código QR		

Tabla 10 RNF06 Autenticidad de los códigos QR

Es parte fundamental del sistema reconocer los códigos QR, además la aplicación debe ser capaz de determinar si dicho código ha sido generado por la aplicación o no, con el fin de brindar seguridad al sistema.

4. Restricciones

Teniendo en cuenta los requerimientos no funcionales del sistema, el alcance del proyecto y las decisiones de implementación, se definieron las siguientes restricciones:

PLATAFORMA WEB:

- El navegador web debe soportar JavaScript. La aplicación web debe funcionar en las últimas versiones de Google Chrome y Firefox.

APLICACIÓN MÓVIL:

- El dispositivo móvil debe tener sistema operativo Android 5.0 o superior.
- El dispositivo móvil debe tener cámara integrada.
- El dispositivo móvil debe tener acceso a internet.

OTROS:

- Los pagos a través de la aplicación deben realizarse por medio de los métodos ofrecidos por el sistema de pagos (i.e tarjeta de crédito, tarjeta débito, transferencia bancaria, entre otros).
- Los eventos creados en la aplicación deben ser de pequeña escala (aproximadamente 100 personas).

5. Casos de uso

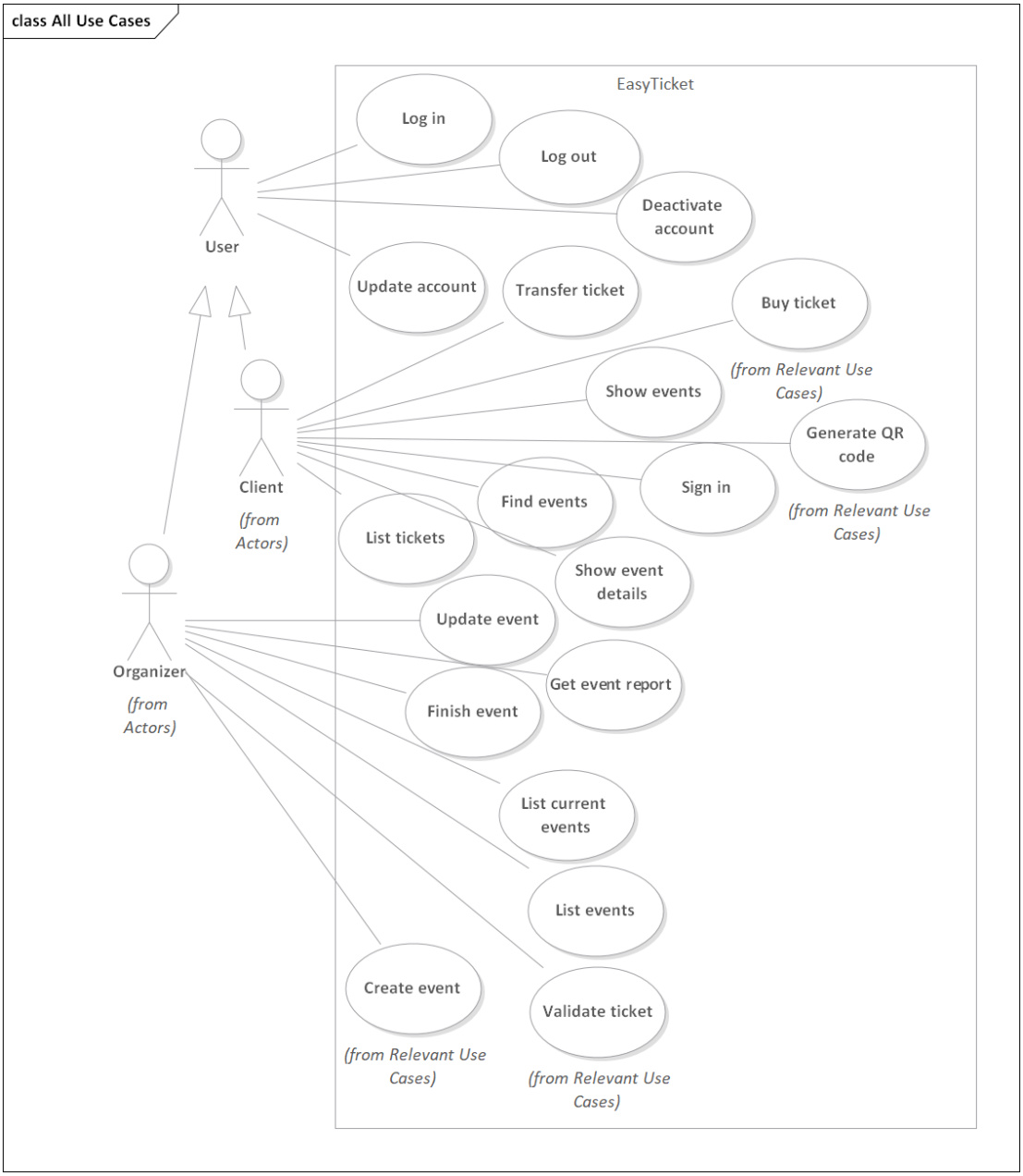

Figura 3 Casos de uso

Dados los requerimientos funcionales del proyecto, descritos en el SRS, en la Figura 3 se muestran los casos de uso del sistema. Los principales casos de uso se encuentran marcados como relevantes, es decir, los más importantes para la toma de decisiones arquitecturales.

En las tablas Tabla 11, Tabla 12, Tabla 13 y Tabla 14 se describen los casos de uso relevantes. Estos se encuentran en mayor detalle en el documento SAD.

ORGANIZER:

Create event	
Descripción	Creación de un evento por parte de un organizador para ser publicado en el sistema
Precondiciones	• El organizador debe contar con una cuenta, creada previamente por el administrador del sistema • El usuario debe iniciar sesión en la plataforma
Postcondiciones	Actualización de los eventos asociados al organizador
Flujo	1. En la página web, ingresar a la sección de eventos 2. En la lista de eventos seleccionar la opción para crear un evento nuevo 3. Diligenciar el formulario con los datos requeridos 4. Enviar el formulario

Tabla 11 Caso de uso Create event

Validate ticket	
Descripción	Validación del código de barras presentado por un cliente al ingresar a un evento
Precondiciones	• El organizador debe contar con un evento activo • El usuario debe iniciar sesión en la aplicación móvil
Postcondiciones	• Modificación del estado del tiquete asociado al cliente • Actualización del evento
Flujo	1. En la aplicación móvil, seleccionar el evento al que se está controlando el acceso 2. Escanear con el dispositivo móvil el código de barras presentado por el cliente (PDF417 o QR) 3. Confirmar la respuesta recibida por el sistema

Tabla 12 Caso de uso Validate ticket

CLIENT:

Buy Ticket	
Descripción	Compra de un tiquete para un evento
Precondiciones	• El cliente debe estar registrado en el sistema
Postcondiciones	• Creación de un tiquete asociada al usuario

Flujo	1. En la página web, seleccionar un evento 2. Seleccionar la opción de compra 3. Llenar el formulario con los datos requeridos para realizar la compra en línea 4. El sistema procesa el pago y envía un correo con la confirmación de la transacción

Tabla 13 Caso de uso Buy ticket

Generate QR	
Descripción	General el código QR de un tiquete existente
Precondiciones	• El cliente debe haber realizado la compra de un tiquete
Postcondiciones	• El usuario obtiene el código QR mediante correo electrónico
Flujo	1. En la plataforma web, ingresar a la sección de tiquetes 2. Seleccionar un tiquete 3. El usuario selecciona la opción para generar el código de barras 4. El sistema envía un correo con el código QR respectivo

Tabla 14 Caso de uso Generate QR

6. Vista lógica

Para el análisis de las principales funcionalidades se hace uso del patrón Entity, Boundary, Control (EBC) con el fin de determinar los elementos que interactúan en cada uno de los casos de uso.

Los subsistemas definidos corresponden a las principales unidades en las que está dividido el sistema:

- EasyTicket Web: plataforma web para clientes y organizadores
- EasyTicket API: responsable de exponer la lógica de negocio
- EasyTicket Mobile: aplicación destinada a la validación de tiquetes por parte de organizadores
- EasyTicket Validation Service: servicio encargado de determinar la validez de un tiquete

De igual manera, se presenta la interacción entre las partes del sistema para el flujo principal de cada funcionalidad en un diagrama de secuencia.

Los sistemas externos que se contemplan en cada uno de los casos de uso incluyen:

- Database: persistencia de la información de la plataforma
- PaymentsSystem: sistema encargado de las transacciones necesarias para la compra de tiquetes
- MailServer: servidor dedicado al envío de correos a los usuarios
- QueueSystem: sistema de comunicación asíncrona, usado en tareas no transaccionales
- NoSQLDatabase: base de datos no relacional, usada para el almacenamiento de información de tiquetes y consultada por el servicio de validación

CREATE EVENT

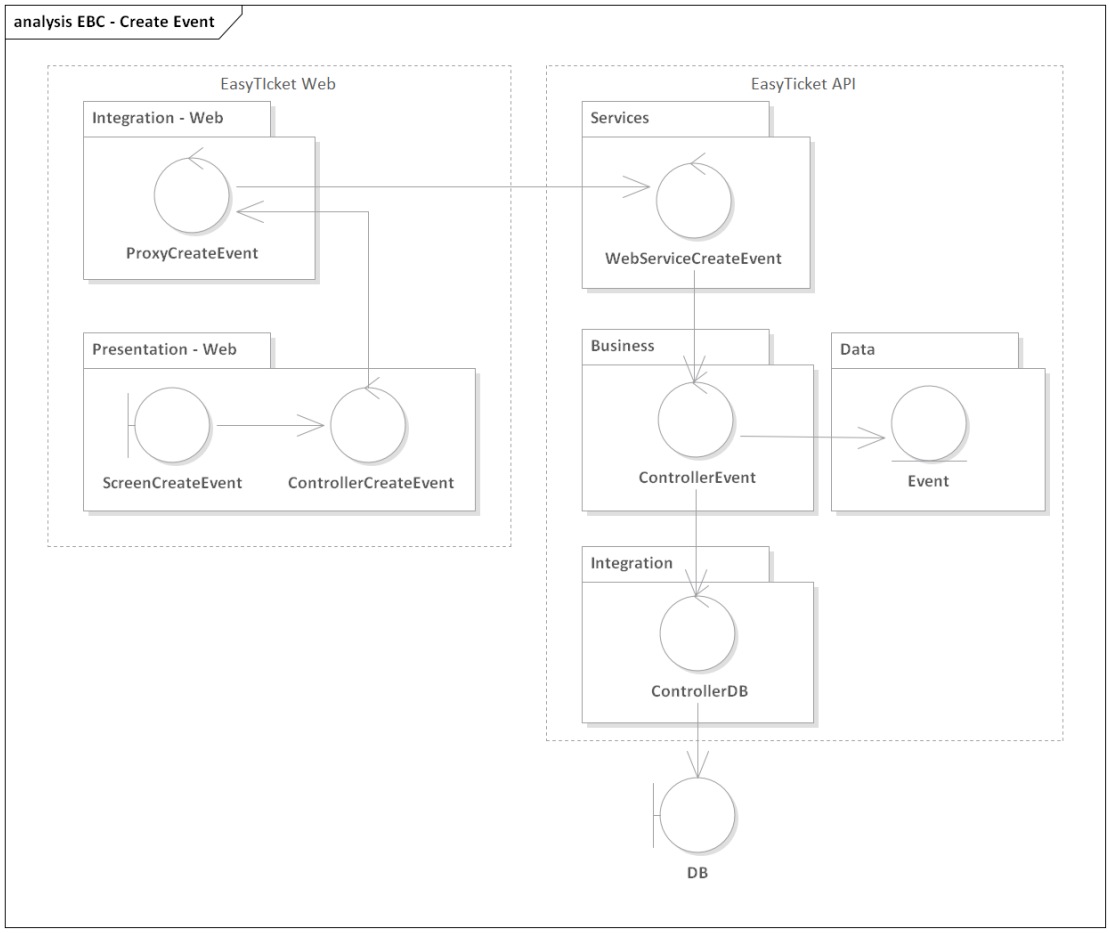

Figura 4 EBC - Create event

En la Figura 4 se ilustran los componentes de la plataforma web y el servidor que interactúan para la creación de un evento. Estos elementos se describen en detalle en la Tabla 15 y Tabla 16.

EasyTicket Web	
Integration Layer	
ProxyCreateEvent	Responsable de enviar las solicitudes al servicio de creación de eventos
Presentation Layer	
ScreenCreateEvent	Interfaz que cuenta con el formulario para el ingreso de datos de un nuevo evento
ControllerCreateEvent	Encargado de procesar los datos del formulario para ser enviados al servicio correspondiente

Tabla 15 EBC - Create event EasyTicket Web

EasyTicket API	
Services Layer	
ServiceCreateEvent	Servicio expuesto para el registro de un nuevo evento
Business Layer	
ControllerEvent	Encargado de la lógica necesaria para la administración de eventos de la aplicación
Data Layer	
Event	Entidad que corresponde a un evento del sistema
Integration Layer	
ControllerDB	Responsable de la integración con el sistema de base de datos

Tabla 16 EBC - Create event EasyTicket API

En la Figura 5 se muestra la interacción entre los componentes descritos anteriormente. El flujo se describe con mayor detalle en el SAD.

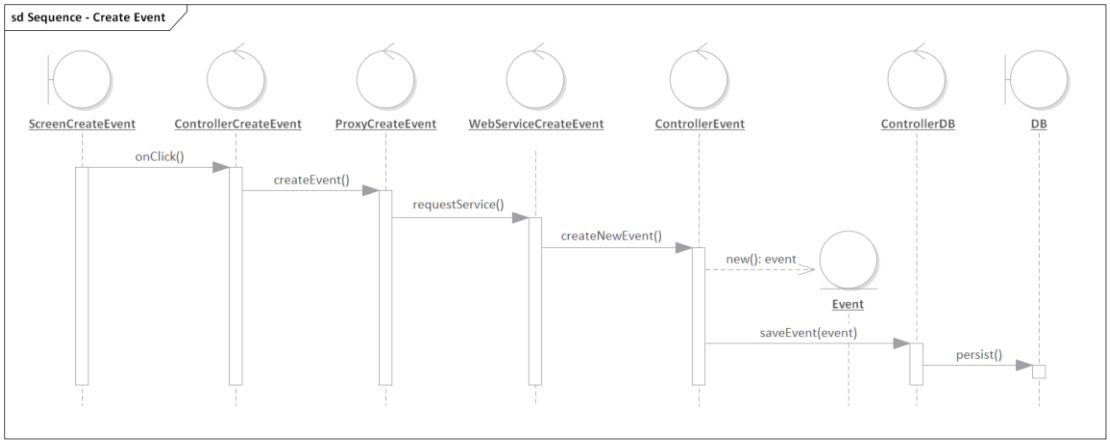

Figura 5 Diagrama de secuencia - Create event

VALIDATE TICKET

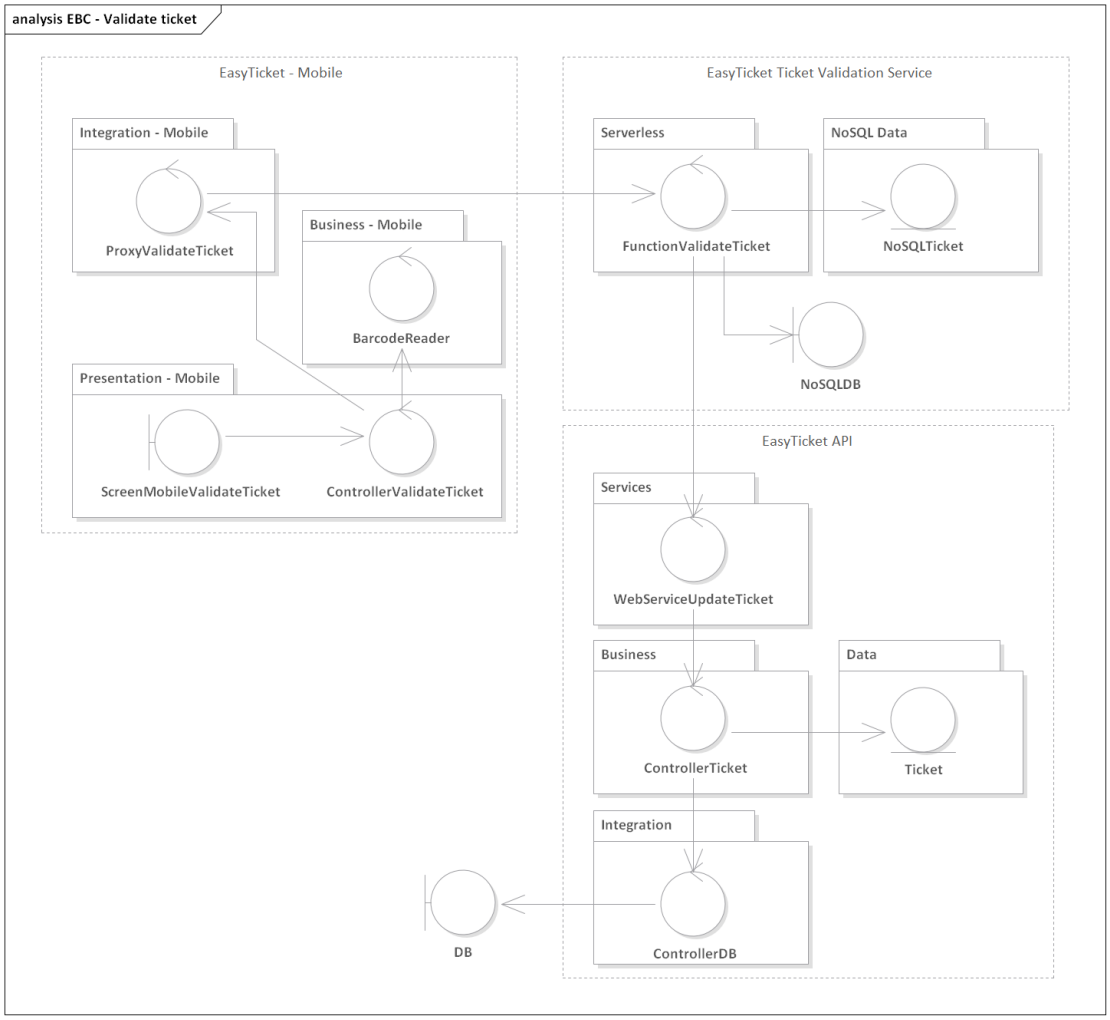

Figura 6 EBC - Validate ticket

En la Tabla 17, Tabla 18 y Tabla 19 se describen los componentes presentados en la Figura 6 para la validación de un tiquete por parte de un organizador desde la aplicación móvil.

EasyTicket Mobile	
Integration Layer	
ProxyValidateTicket	Encargado de solicitar el servicio de validación de tiquetes
Business Layer	
BarcodeReader	Componente responsable de obtener la información requerida del código de barras escaneado
Presentation Layer	
ScreenValidateTicket	Pantalla en la que se realiza la lectura del código de barras

ControllerValidate	Responsable de preparar los datos para ser enviados al servicio de validación

Tabla 17 Validate ticket EasyTicket Mobile

EasyTicket Validation Service	
Serverless Layer	
FunctionValidate	Función encargada de determinar la validez de un tiquete presentado
Data Layer	
NoSQLTicket	Representa un tiquete del sistema. Almacenado en una base de datos no relacional

Tabla 18 Validate ticket EasyTicket Validation Service

EasyTicket API	
Services Layer	
ServiceUpdateTicket	Servicio responsable de la actualización de un tiquete
Business Layer	
ControllerTicket	Administración de los tiquetes en el sistema
Data Layer	
Ticket	Entidad que corresponde a un tiquete del sistema
Integration Layer	
ControllerDB	Responsable de la integración con el sistema de base de datos

Tabla 19 Validate ticket EasyTicket API

En la Figura 7 se muestra la interacción entre los componentes descritos anteriormente. El flujo se describe con mayor detalle en el SAD.

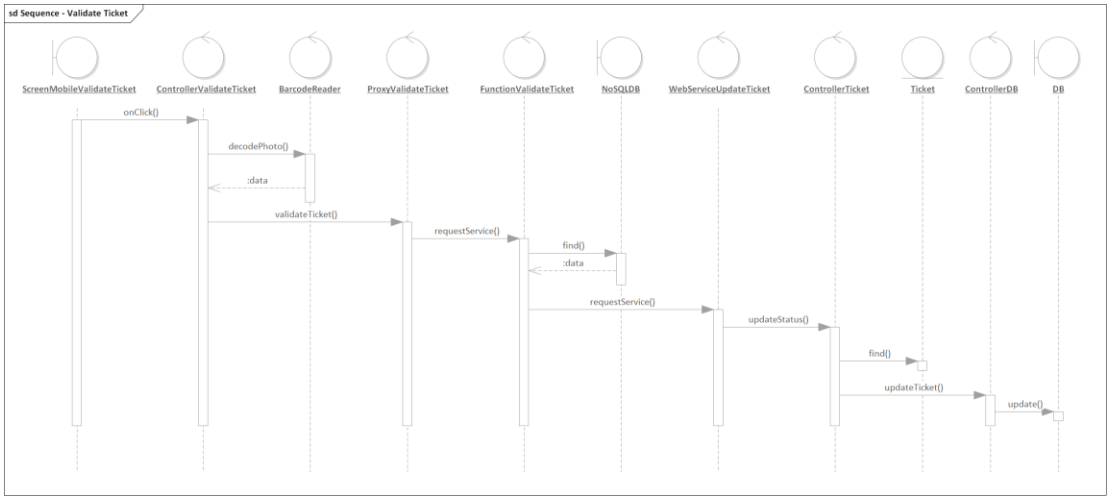

Figura 7 Diagrama de secuencia - Validate ticket

BUY TICKET

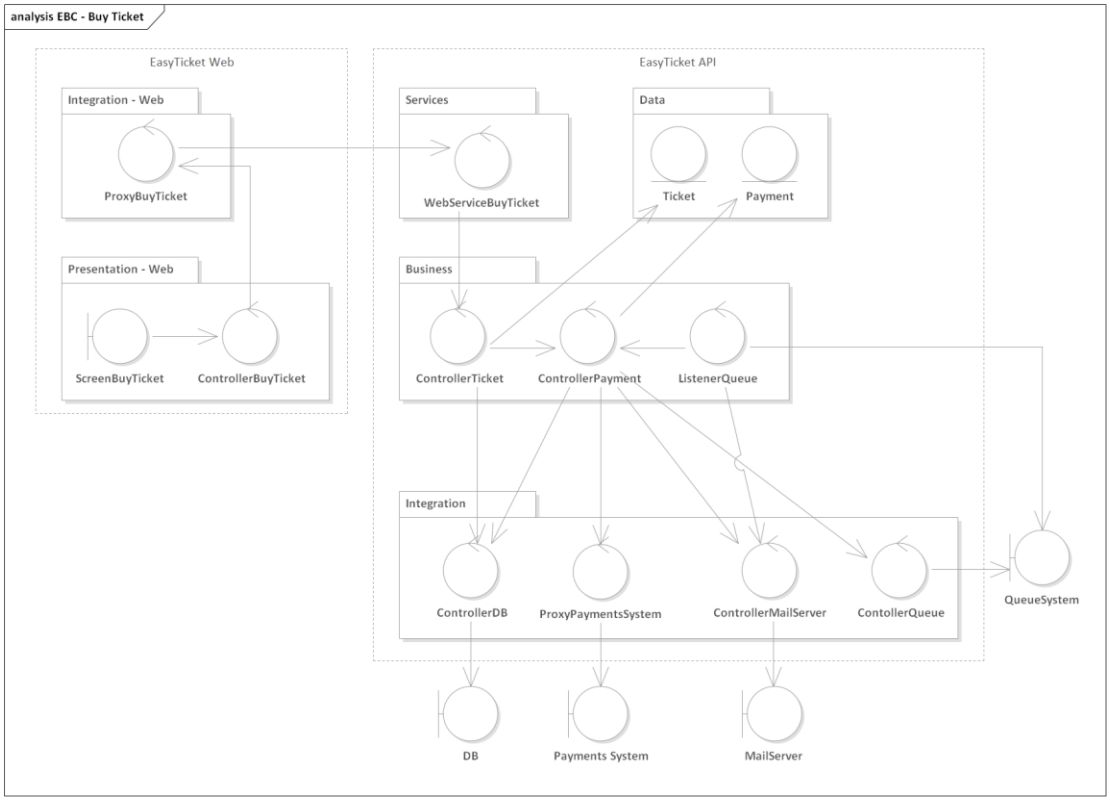

Figura 8 EBC - Buy ticket

Los componentes necesarios para la compra de un tiquete desde la plataforma web se presentan en la Figura 8, y se describen en la Tabla 20 y Tabla 21.

EasyTicket Web	
Integration Layer	
ProxyBuyTicket	Consume el servicio de compra de tiquetes
Presentation Layer	
ScreenBuyTicket	Pantalla que exhibe información de un evento específico y los tiquetes disponibles para la compra
ControllerBuyTicket	Prepara los datos seleccionados por el usuario para solicitar el servicio de compra

Tabla 20 Buy ticket EasyTicket Web

EasyTicket API	
Services Layer	
ServiceBuyTicket	Servicio expuesto para la compra de tiquetes

Business Layer	
ControllerTicket	Administración de los tiquetes en el sistema
ControllerPayment	Manejo de los pagos realizados en la aplicación
ListenerQueue	Componente que responde a mensajes asíncronos
Data Layer	
Ticket	Entidad que corresponde a un tiquete del sistema
Payment	Entidad que representa un pago
Integration Layer	
ControllerDB	Responsable de la integración con el sistema de base de datos
ControllerPayments	Encargado de la comunicación con el sistema de pagos
ControllerMailServer	Solicita los servicios al servidor de correos
ControllerQueue	Maneja la integración con la cola usada para mensajería asíncrona

Tabla 21 Buy ticket EasyTicket API

En la Figura 7 se muestra la interacción entre los componentes descritos anteriormente. El flujo se describe con mayor detalle en el SAD.

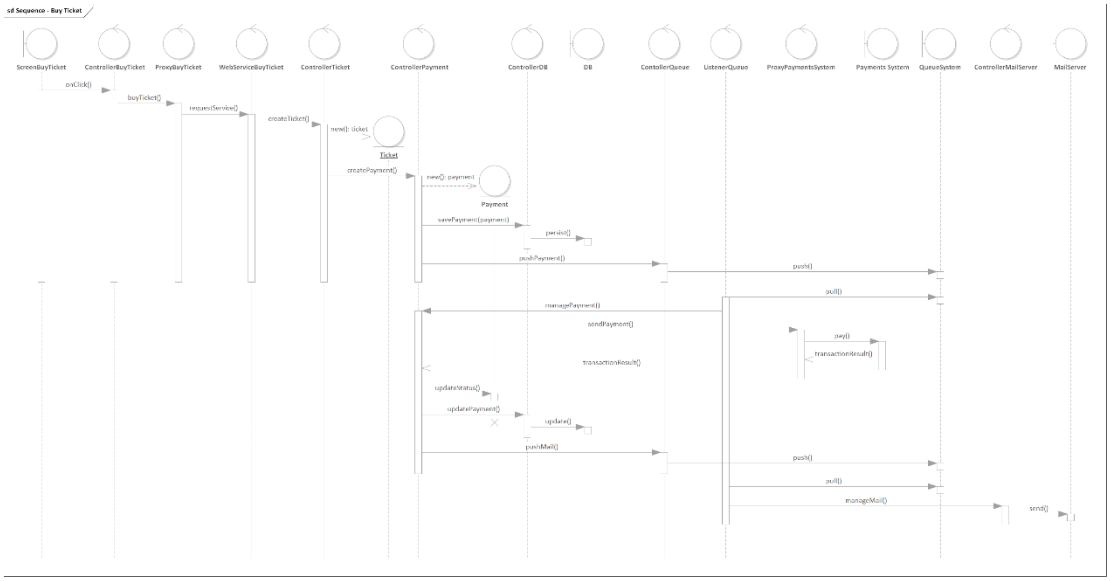

Figura 9 Diagrama de secuencia - Buy ticket

GENERATE QR

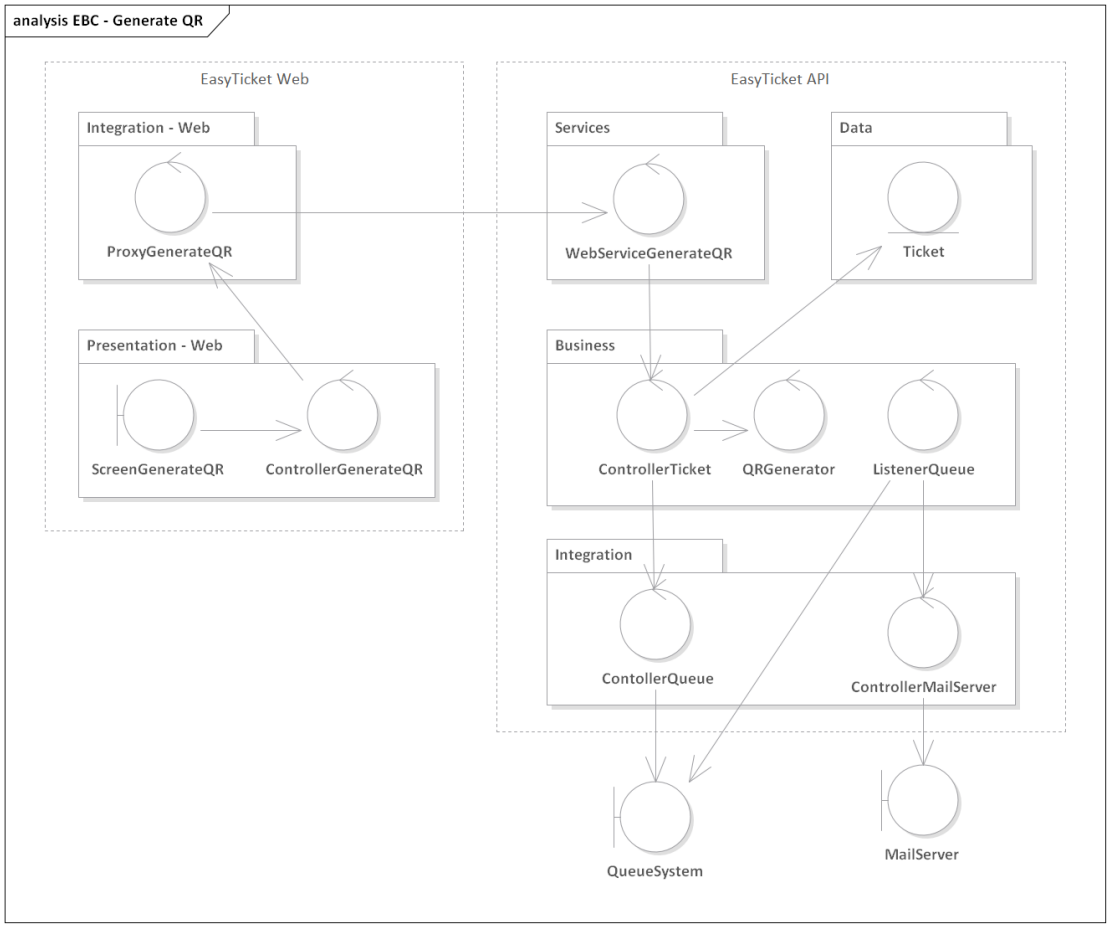

Figura 10 EBC - Generate QR

La estructura de lo componentes para la creación del código QR de un tiquete se muestra en la Figura 10, de igual manera, en la Tabla 22 y Tabla 23 se describen en detalle estos elementos.

EasyTicket Web	
Integration Layer	
ProxyGenerateQR	Consume el servicio de compra de tiquetes
Presentation Layer	
ScreenGenerateQR	Pantalla que exhibe información de un evento específico y los tiquetes disponibles para la compra
ControllerGenerateQR	Prepara los datos seleccionados por el usuario para solicitar el servicio de compra

Tabla 22 Generate QR EasyTicket Web

EasyTicket API	
Services Layer	
ServiceGenerateQR	Servicio expuesto para generar el código QR de un tiquete
Business Layer	
ControllerTicket	Administración de los tiquetes en el sistema
ListenerQueue	
Data Layer	
Ticket	Entidad que corresponde a un tiquete del sistema
Integration Layer	
ControllerMailSer-ver	Solicita los servicios al servidor de correos
ControllerQueue	Maneja la integración con la cola usada para mensajería asíncrona

Tabla 23 Generate QR EasyTicket API

En la Figura 11Figura 7 se muestra la interacción entre los componentes descritos anteriormente. El flujo se describe con mayor detalle en el SAD.

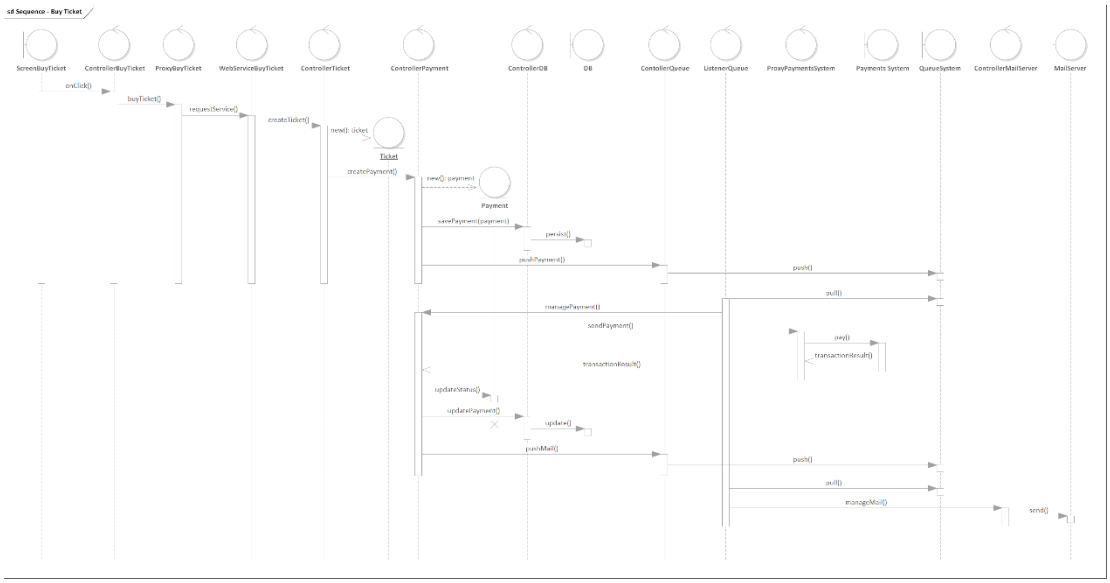

Figura 11 Diagrama de secuencia - Generate QR

7. Vista de implementación

En la Figura 12 se presenta la agrupación en componentes de los EBC detallados anteriormente, con el objetivo de exponer las dependencias entre estos y la modularidad del sistema en funcionalidades específicas de negocio. Estos componentes se agrupan en capas, que representan divisiones en los intereses de los elementos de cada subsistema.

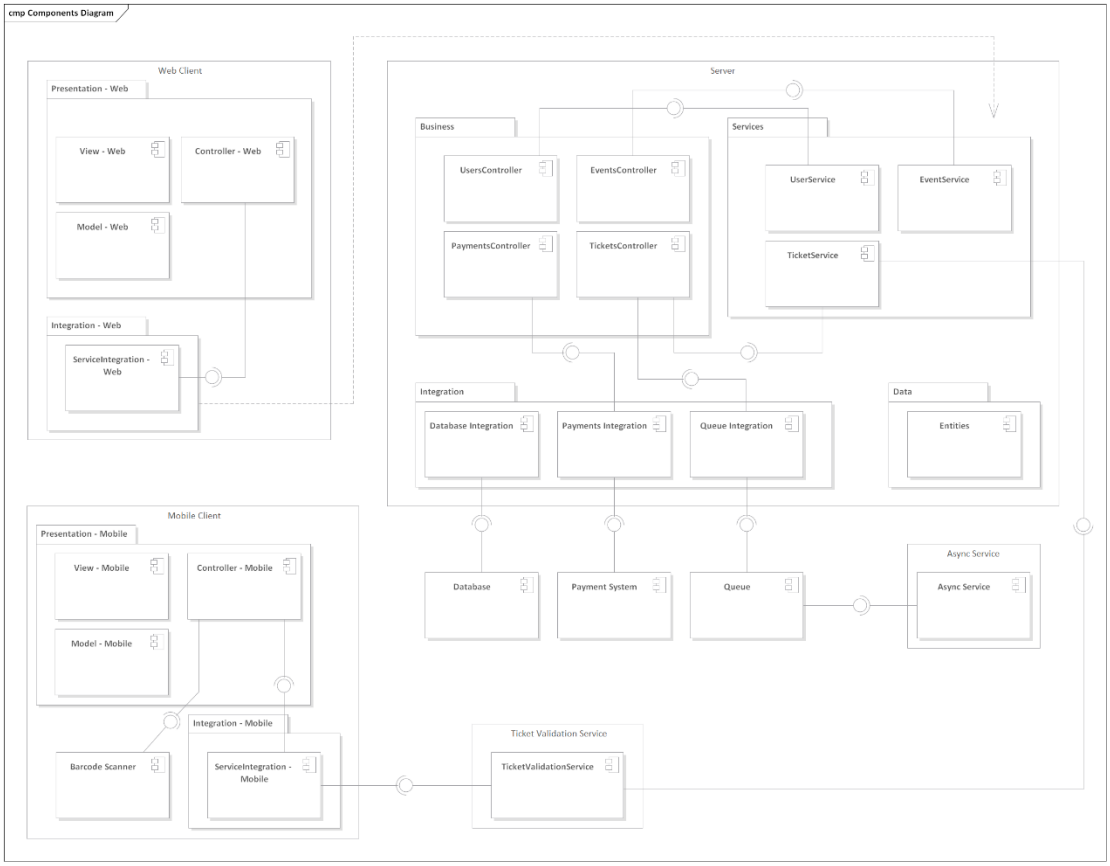

Figura 12 Componentes del sistema

En la Tabla 24, Tabla 25, Tabla 26 y Tabla 27 presentadas a continuación se listan las capas y los componentes que las constituyen, para cada de los subsistemas que hacen parte de la solución.

Web Client	
Presentation - Web	Paquete que agrupa los componentes necesarios para la presentación del cliente web

View - Web	Componente encargado de manejar la interfaz de usuario de la plataforma web
Model - Web	Componente que contiene una representación de los datos que maneja el sistema y su lógica de negocio
Controller - Web	Componente encargado de la comunicación entre el componente Model - Web y el componente View - Web, facilitando el flujo de información entre ellos
Integration - Web	Paquete que agrupa los componentes necesarios para la integración del cliente web con el api del sistema
ServiceIntegration - Web	Componente encargado de facilitar la integración del cliente web con el API del sistema. Este componente expone funcionalidades que son consumidas por el componente Controller - Web

Tabla 24 Componentes Web Client

Mobile Client	
Presentation - Mobile	Paquete que agrupa los componentes necesarios para la presentación del cliente móvil
View - Mobile	Componente encargado de manejar la interfaz de usuario de la aplicación móvil
Model - Mobile	Componente que contiene una representación de los datos que maneja el sistema y su lógica de negocio
Controller - Mobile	Componente encargado de la comunicación entre el componente Model - Mobile y el componente View - Mobile, facilitando el flujo de información entre ellos.
Integration - Mobile	Paquete que agrupa los componentes necesarios para la integración del cliente móvil con el api del sistema y otros servicios
ServiceIntegration - Mobile	Componente encargado de facilitar la integración del cliente móvil con el API del sistema y otros servicios. Este componente expone funcionalidades que son consumidas por el componente Controller - Mobile
Barcode Scanner	Componente encargado de realizar funciones que requieran el escaneo de códigos de barras. Este componente expone funcionalidades que son consumidas por el componente Controller - Mobile

Tabla 25 Componentes Mobile Client

Server	
Business	Paquete que agrupa los componentes que contienen la lógica de negocio del sistema
UsersController	Componente encargado de manejar las funcionalidades relacionadas con los usuarios. Este componente expone funcionalidades que son consumidas por por el componente UserService

EventsController	Componente encargado de manejar las funcionalidades relacionadas con los eventos. Este componente expone funcionalidades que son consumidas por por el componente EventService
PaymentsController	Componente encargado de manejar las funcionalidades relacionadas con los pagos.
TicketsController	Componente encargado de manejar las funcionalidades relacionadas con los tiquetes. Este componente expone funcionalidades que son consumidas por por el componente TicketService
Services	Paquete que agrupa los componentes que exponen parte de la lógica de negocio por medio de servicios web
UserService	Componente encargado de exponer los servicios relacionados con los usuarios
EventService	Componente encargado de exponer los servicios relacionados con los eventos
TicketService	Componente encargado de exponer los servicios relacionados con los tiquetes
Integration	Paquete que agrupa los componentes que facilitan la integración del api con sistemas externos
Database Integration	Componente encargado de realizar la integración del sistema con la base de datos
Payments Integration	Componente encargado de realizar la integración del sistema con el sistema de pagos. Este componente expone funcionalidades que son consumidas por por el componente PaymentsController
Queue Integration	Componente encargado de realizar la integración del sistema con la cola. Este componente expone funcionalidades que son consumidas por por el componente TicketsController
Data	Paquete que agrupa los componentes que representan los objetos del sistema
Entities	Componente que contiene las entidades del sistema

Tabla 26 Componentes Server

Ticket Validation Service	
TicketValidationService	Componente encargado de realizar funciones relacionadas con la validación de tiquetes. Este componente expone funcionalidades que son consumidas por el componente ServiceIntegration - Mobile

Tabla 27 Componentes Ticket Validation Service

V- DISEÑO DE LA SOLUCIÓN

Teniendo en cuenta la solución planteada y el análisis arquitectural realizado anteriormente, en este capítulo se pretende presentar de manera general el diseño de la solución, así como describir las herramientas y tecnologías utilizadas para la implementación del proyecto.

1. Presentación de tecnologías

Teniendo en cuenta la solución planteada en secciones anteriores del documento, EasyTicket se definió en tres subsistemas: portal web, aplicación móvil y servidor central. Por consiguiente, fue necesario elegir diferentes plataformas y/o tecnologías con las que se puedan suplir satisfactoriamente las exigencias de cada componente, junto a las necesidades del equipo de desarrollo. Para seleccionar las tecnologías se contemplaron factores como el conocimiento de los integrantes, su facilidad de integración con otros sistemas, entre otros.

Se tuvo en cuenta proyectos como RealWorld, el cual provee diversos ejemplos de aplicaciones que se adhieren a una misma especificación, cada una desarrollada en una plataforma o lenguaje diferente, con el fin de ofrecer un conjunto de demostraciones que le brinden a los desarrolladores de software una guía para desarrollar aplicaciones full-stack en las plataformas más populares [23], [24].

Aplicación móvil

Para la aplicación móvil se eligió Android como la plataforma objetivo, puesto que su peso en el mercado es mayor y la disponibilidad de dispositivos en el inventario del equipo de desarrollo la favorece, frente a la segunda alternativa, iOS, sistema que limita el despliegue únicamente desde dispositivos Apple [25].

NativeScript se eligió como subplataforma de desarrollo de aplicaciones, gracias a su facilidad de integración con el sistema nativo de Android, y además, las librerías utilizadas para la lectura de los códigos de barras [25].

Las otras alternativas incluyen React-Native, sin embargo, teniendo en cuenta la experiencia y el conocimiento del grupo, fue descartada debido al tiempo adicional que implicaría la capacitación en esta tecnología. Por el contrario, la integración con Angular de NativeScript favorece al equipo de desarrollo al tener en cuenta la experiencia en proyectos anteriores.

Portal web

Para el portal web se decidió hacer una aplicación de una sola página (SPA), debido a que es la tendencia actual del desarrollo web, además, es el enfoque del front-end en los ejemplos de RealWorld. Entre los frameworks más utilizados expuestos en RealWorld encontramos React, Angular, Elm y Vue, cada uno ofrece diferentes maneras de desarrollar, sin embargo, teniendo en cuenta el conocimiento del equipo de desarrollo y su experiencia, Angular fue considerada la mejor opción.

El tiempo de capacitación en la plataforma seleccionada sería el menor comparada con las demás. En la parte técnica, todas las tecnologías puestas a consideración contaban con las mismas capacidades de integración y de desarrollo de aplicaciones, pues todas están escritas sobre JavaScript, lenguaje que la mayoría de navegadores modernos implementa según el estándar ECMAScript 5.

Al momento de desarrollar, todas las plataformas ofrecen un entorno de desarrollo basado en JavaScript y TypeScript, a excepción de Elm, que requiere utilizar su propio lenguaje. Este fue otro factor considerado para la elección de la plataforma, debido a que los integrantes del grupo enfocados en el desarrollo del portal web contaban con experiencia en dicho entorno.

Servidor central

Al igual que en los otros subsistemas, se tuvo en cuenta la tendencia de desarrollo web y móvil para este componente del proyecto. Teniendo en cuenta las plataformas enfocadas en back-end de RealWorld, encontramos diferentes alternativas como Django, Node, Laravel y ASP.NET. De la misma manera que en el portal web, el conocimiento de los desarrolladores tuvo una gran influencia en la elección de la plataforma. Asimismo, teniendo en cuenta el objetivo de utilizar una plataforma sin servidor, las opciones ofrecidas por Amazon AWS Lambda y Google Cloud Functions, limitaban los lenguajes disponibles para desarrollar las aplicaciones, dejando únicamente disponibles a Django y a Node.

La experiencia de los desarrolladores utilizando estas plataformas favoreció a Amazon AWS frente a Google Cloud, y a Django frente a Node. Por lo tanto, el proyecto se decidió implementar en estas elecciones.

2. Modelo de datos

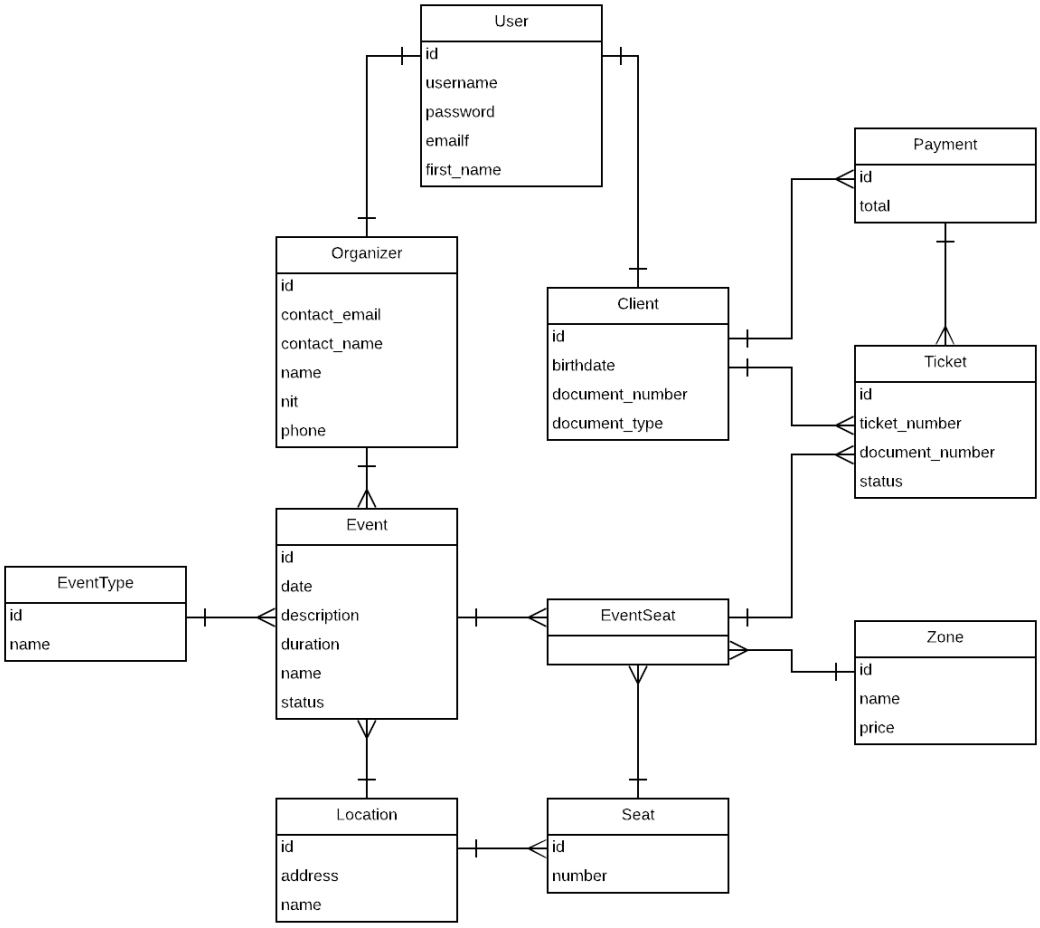

Figura 13 Modelo de datos

En la Figura 13 se observa el diagrama de datos definido para modelar los objetos del sistema. Para ver en detalle la descripción del modelo de datos, revisar el documento SDD.

3. Infraestructura

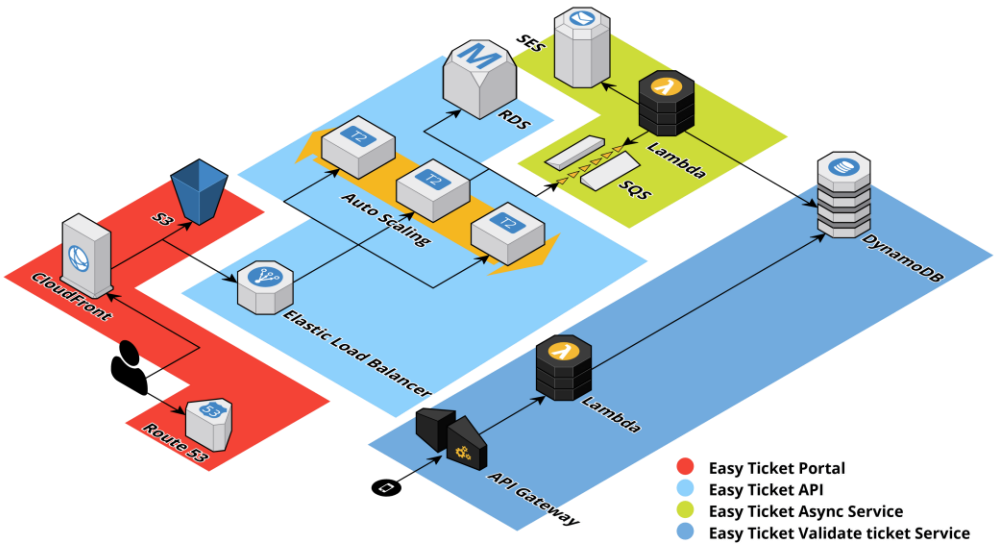

Figura 14 Diagrama de infraestructura

En la Figura 14, se muestra la infraestructura del sistema modelando los componentes de AWS que se desplegaron haciendo uso de la herramienta Terraform. El sistema se compone de cuatro principales módulos que son:

“Easy Ticket Portal” se compone de:

- Route 53: servicio de DNS. Encargado de buscar el dominio y encontrar la distribución más cercana de la página.
- CloudFront: servicio de distribución. Encargado de distribuir todo el contenido estático por la red de amazon.
- S3: servicio de almacenamiento de archivos. Encargado de almacenar los archivos estáticos(html/css/js).

“Easy Ticket API” se compone de:

- Elastic Load Balancer: balanceador de cargas. Encargado de verificar la mejor instancia para asignarle una petición entrante.
- Instancias Fargate: tareas donde se ejecuta el servicio principal, esta corre la imagen Docker que se desplegó previamente con el código de los servicios.
- RDS: servicio de base de datos relacional.

- SQS: servicio de cola. En este servicio se depositarán los mensajes para posteriormente ser procesados por el servicio asíncrono, por ejemplo: correos, conexión con sistema de pagos, entre otros.

"Easy Ticket Async Service" se compone de:

- SQS: servicio de cola.
- Lambda: servicio serverless que será encargado de procesar los mensajes que lleguen a la cola.
- SES: servicio de correos.
- DynamoDB: servicio de base de datos no relacional.

"Easy Ticket Validate ticket service" se compone de:

- API Gateway: expondrá el Lambda como API.
- Lambda: servicio serverless que será encargado de procesar las peticiones de validación y confirmar la información en la base de datos.
- DynamoDB: servicio de base de datos no relacional.

A continuación, se explica brevemente la interacción de los diferentes componentes con el caso de uso de Comprar Boleta y Validar Boleta.

En el componente de "Easy Ticket Portal", el usuario ingresa al dominio https://simpleticket.co/ donde el servicio de Route 53 busca el dominio y procede a encontrar el servidor más cercano que contenga la página web, se obtienen los archivos estáticos (html, css y js) y luego se presenta la opción de compra.

En el componente de "Easy Ticket API", se verifica cual es la mejor instancia para recibir la petición, valida la información recibida y crea el pago en la base de datos con un estado de "PENDING", posteriormente se agrupa la información del pago para colocar el mensaje en la cola.

El componente "Easy Ticket Async Service" recibe la información del pago desde la cola, se procesa la información y se envía al sistema de pagos, luego recibe la respuesta de dicho servicio y se notifica para cambiar el estado del pago a "ACCEPTED" o "REJECTED".

Finalmente, en la aplicación móvil del componente "Easy Ticket Validate ticket service" se selecciona qué tipo de código se va a escanear (QR o PDF417) se procesa la información y se envía al servicio de validación expuesto por el API Gateway, luego se

valida dicha información en la base de datos de DynamoDB y notifica la respuesta correspondiente.

VI- DESARROLLO DE LA SOLUCIÓN

Teniendo en cuenta el análisis arquitectural y el diseño del sistema, en este capítulo se pretende presentar el proceso de desarrollo llevado a cabo, así como presentar el producto final.

1. Metodología

En esta sección se resume la metodología utilizada para desarrollar el proyecto. Consiste principalmente en cuatro fases: inicio, elaboración, construcción y resultados. Se definieron estas etapas teniendo en cuenta las propuestas de AUP (Agile Unified Process), no obstante, la etapa de transición se deja por fuera del alcance del proyecto, por lo que se omitió. Siguiendo un enfoque de desarrollo ágil, también se tienen en cuenta algunos elementos de la metodología Scrum, que serán detallados en la fase de elaboración y construcción.

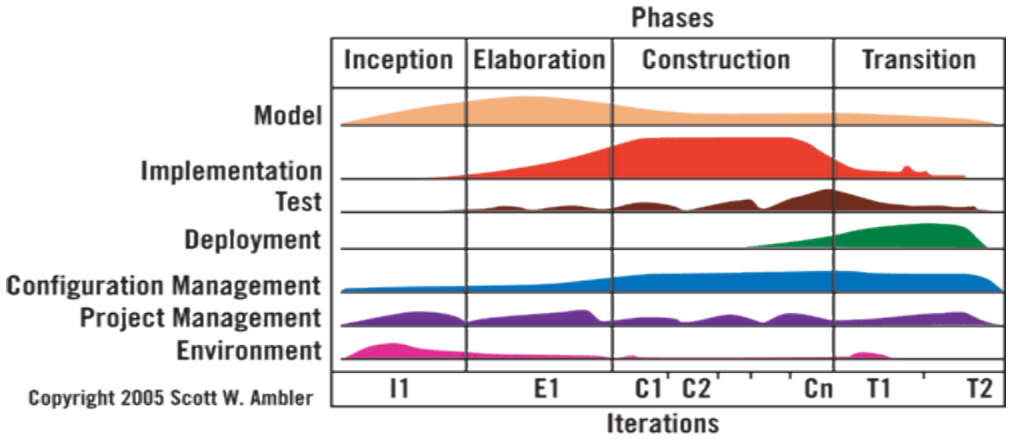

Figura 15 Fases del AUP, tomada de [26]

Fase de inicio

La primera parte del proyecto consiste en identificar el problema, el cual se manifestó a raíz de los principales sistemas de venta de boletería. Después de haber definido la problemática, se presentó una propuesta que busca solucionar algunos de los inconvenientes identificado, a través del uso de las tecnologías modernas. Esta propuesta

consiste en el conjunto de requisitos del proyecto, herramientas a utilizar, metodologías, análisis de riesgos, pruebas y validación.

Además, se definió una serie de actividades que aportaría al proyecto con más fuentes de información. Esto consistió en explorar el problema, lo que comprende el entendimiento de la situación actual de la venta de boletería. Asimismo, el planteamiento de la solución consistió en definir qué busca la solución en esta problemática, es decir, cuáles son sus objetivos.

El plan de proyecto también se desarrolló en esta fase, con el fin de poder predecir el tiempo que podría tomar su desarrollo, y con esto, definir un alcance que pueda satisfacer las necesidades de la solución, sin dejar atrás las capacidades del equipo.

Al final de esta fase, se obtuvo todos los entregables pactados, es decir, especificación de requerimientos del proyecto, un plan de desarrollo, una metodología y documentos que definan superficialmente la solución al problema, mediante las herramientas pactadas.

Fase de elaboración

Utilizando la metodología Scrum, se propuso iteraciones (sprints) de 2 semanas, en las que se desarrolló tanto la implementación de la solución, como los documentos que proporcionaban información acerca de las actividades realizadas. Se definió la arquitectura de alto nivel del sistema, en la que se detallan los principales componentes y la comunicación entre cada uno de estos. También se propuso un plan de pruebas preliminar, con el fin de verificar y validar la calidad y el funcionamiento de los desarrollos propuestos.

De esta fase se obtuvo el documento con la descripción del diseño de software (SDD), el cual contó con la aprobación del director del proyecto de grado. El documento de pruebas desarrollado en esta fase planteó el diseño de las mismas, junto a un plan de aceptación que esperaba ser aplicado en la siguiente fase.

Fase de construcción

En la fase de construcción se esperó implementar la solución partiendo del diseño desarrollado en las anteriores fases, junto a las actividades de verificación y validación relacionadas con el plan de pruebas. Las iteraciones de dos semanas continúan en este punto, en las que en cada avance significativo del proyecto, requería una reunión con el director de grado, con el fin de obtener retroalimentación sobre lo que se estaba desarrollando.

Fase de resultados

En esta fase se obtienen entregables como el prototipo funcional del sistema, documentos de pruebas realizadas en las etapas anteriores, un manual de usuario, y resultados obtenidos en alguna aplicación de la plataforma. Al culminar esta fase, se espera haber cumplido satisfactoriamente con los objetivos planteados y dar por hecho que el sistema funciona y es confiable.

2. Producto final

En esta sección se presentan algunas de las vistas de las aplicaciones realizadas: plataforma web y aplicación móvil. La guía de uso se encuentra en el documento Manual de Uso.

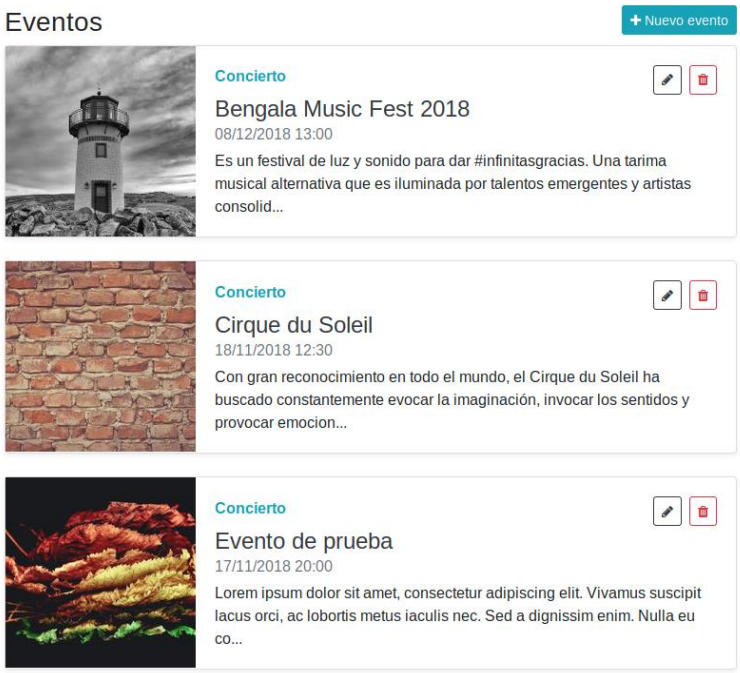

Figura 16 Pantalla con la lista de eventos del organizador

Una vez iniciada sesión como organizador, se listan los eventos creados por el usuario, como se muestra en la Figura 16. En cada tarjeta se muestra la información básica del evento y botones para editar o desactivar el mismo.

Figura 17 Formulario para la creación de eventos

En el formulario de creación de evento, se solicitan los datos necesarios, al igual que la información de localidades para la generación de tiquetes. Una vez creado el evento, este ya es visible en la vista anterior. Este formulario se puede observar en la Figura 17.

Figura 18 Formulario para la compra de un tiquete

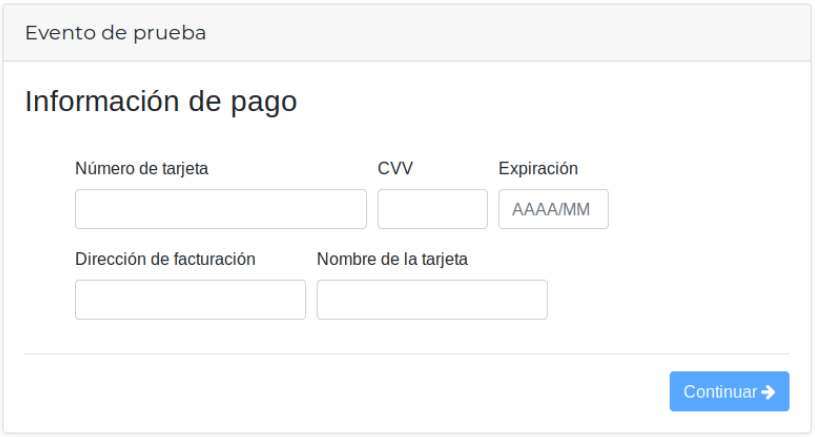

Figura 19 Formulario de pago

Después de haber seleccionado un evento para la compra de tiquetes, el cliente debe diligenciar los formularios requeridos, como se muestra en las figuras Figura 18 y Figura 19. Al finalizar el pago, se enviará al usuario un correo con el código QR asociado al tiquete.

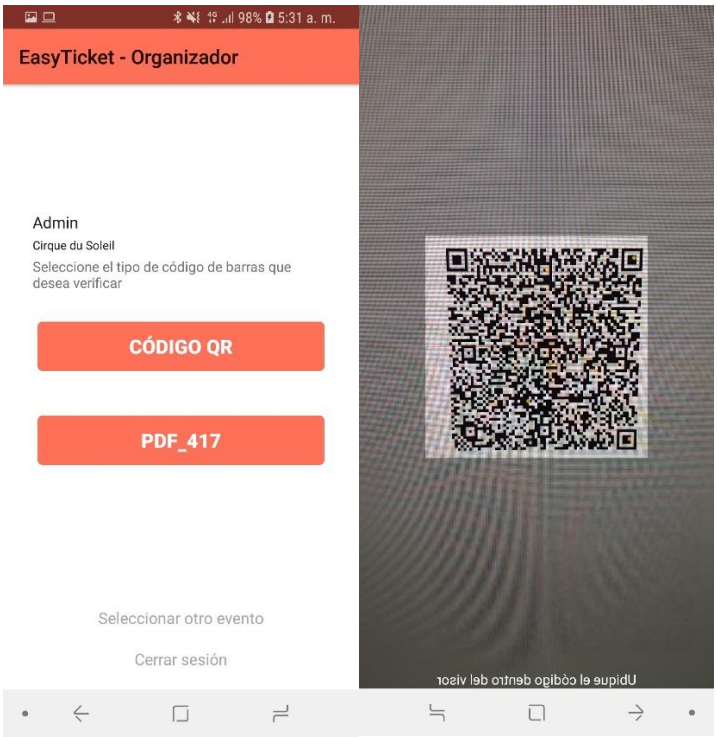

Figura 20 Pantalla de validación de tiquete

Al momento de realizar el control de acceso a un evento, el organizador debe seleccionar el evento deseado y, posteriormente seleccionar el tipo de código a escanear. Esta funcionalidad se puede ver en la Figura 20.

VII- RESULTADOS

En la presente sección, se presentan los resultados obtenidos a lo largo del proceso de desarrollo. Los resultados corresponden a la aceptación y validación del diseño arquitectural del sistema, la experiencia de usuario y la garantía de la correcta ejecución de los componentes.

1. Resultados de pruebas

Las pruebas unitarias y de integración realizadas se adaptan al modelo de ciclo de vida de desarrollo del proyecto, en el cual, las pruebas se definen antes de la implementación de cada componente. En el documento de pruebas se encuentra el diseño de estas con más detalle.

El framework utilizado para la implementación del servidor (Django), incluye clases utilitarias para realizar validaciones de los servicios REST, las cuales fueron utilizadas para probar dichos servicios. Adicionalmente, se utilizaron mocks para realizar pruebas de la comunicación del sistema con los componentes AWS, en donde se simularon las entradas esperadas.

Para realizar las pruebas a la plataforma web, se decidió utilizar dos frameworks: Karma, la cual es una herramienta de pruebas que utiliza un servidor web para la ejecución de plataformas web y Jasmine, un framework de desarrollo guiado por comportamiento (BDD), encargado de realizar pruebas a aplicaciones en JavaScript.

Finalmente, se realizó una prueba piloto en un ambiente controlado por el equipo de trabajo, en el que se simula un evento. Cabe aclarar que el prototipo no fue entregado a un tercero para la realización de la prueba en cuestión, debido a que las funcionalidades especificadas en el alcance del proyecto no eran suficientes para atender todas las necesidades que presenta un evento. En la Figura 21, se observa el proceso llevado a cabo para la realización de la prueba piloto.

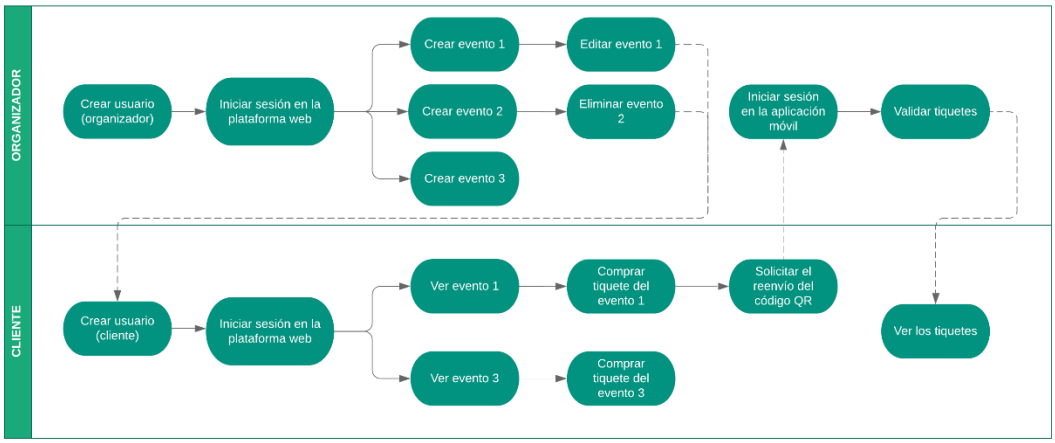

Figura 21 Proceso de prueba piloto

Durante la ejecución de la prueba, el sistema se comportó de manera esperada realizando las funcionalidades solicitadas. Teniendo en cuenta los resultados obtenidos, se concluye que el sistema cumple de manera exitosa los 4 casos de uso arquitecturalmente relevantes planteados en la memoria del proyecto.

2. Validación de usuarios

Para realizar la validación del sistema según la experiencia de uso de los usuarios asistentes a eventos, se aplicó el Technology Acceptance Model con algunas modificaciones. También se realizó la validación de un usuario organizador teniendo en cuenta las características que le puede ofrecer la aplicación.

A continuación, se presentan las variables a evaluar tomadas del TAM, con sus respectivas descripciones:

- Utilidad percibida: expectativa de mejorar la experiencia de los usuarios a la hora de comprar boletería de eventos con EasyTicket.
- Facilidad percibida de uso: expectativa de los usuarios de que EasyTicket sea fácil de usar e intuitivo.
- Compatibilidad: percepción en cuanto a la consistencia de EasyTicket con el espacio o ambiente en donde se implemente.
- Comportamiento de intención de uso: intención de los usuarios de usar EasyTicket.

- Compra de boleta: acción influenciada por un comportamiento positivo de intención de uso se EasyTicket

Teniendo en cuenta los resultados obtenidos por medio de los cuestionarios, se llegan a las siguientes conclusiones:

- La interfaz de EasyTicket es intuitiva y clara en las funcionalidades que ofrece.
- Los participantes demuestran buena disposición hacia la aplicación y están dispuestos a usarla.
- Los participantes encuentran útil la aplicación para realizar la compra de tiquetes de eventos de su interés.

En el documento TAM, se presenta el perfil de los participantes además del proceso realizado y conclusiones adicionales.

3. Validación de expertos

Se realizó la validación arquitectural del sistema por medio del método Architecture Tradeoff Analysis Method (ATAM), el cual busca identificar las consecuencias de las decisiones arquitecturales a luz de los requerimientos no funcionales. Para ello, se contó con la participación de dos expertos a quienes se les presentaron las decisiones arquitecturales que definieron la arquitectura del sistema. Algunos de los comentarios más significativos incluyen:

- Combinación de estilos arquitecturales: las ventajas presentadas son la independencia entre módulos, lo que disminuye el acoplamiento entre los mismos y la creación de puntos únicos de fallo. La principal desventaja identificada hace referencia al monitoreo del sistema dada la implementación de múltiples estilos arquitecturales.

- Servicio de validación de tiquetes: se reconoció como principal ventaja la disminución de la carga de los servicios expuestos por el servidor principal. El riesgo más relevante identificado es el ataque a este servicio, lo que puede terminar en una denegación de servicios.

- Servicio para tareas asíncronas: apoya la eficiencia del sistema puesto que disminuye la saturación de los servicios y permite el procesamiento de múltiples tareas de forma rápida y eficiente. Sin embargo, se debe tener cuidado con el manejo de mensajes de confirmación o rechazo de transacciones, para evitar procesar un mensaje múltiples veces.

Se evidenció que los expertos tienen una percepción positiva de la arquitectura planteada, puesto que los estilos adoptados buscan apoyar los atributos de calidad definidos para cada componente del sistema.

En el documento ATAM, se presenta el perfil de los expertos además del proceso realizado para la validación de la arquitectura.

VIII- CONCLUSIONES

1. Análisis de impacto

El presente proyecto de grado ofrece un ejemplo útil para promocionar el uso de nuevas tecnologías y arquitecturas en la ingeniería de software. Esto se ve reflejado en los nuevos desarrollos basados en microservicios, incluyendo este proyecto, dejando atrás los sistemas monolitos o arquitecturas cliente-servidor clásicas. Se brinda la posibilidad de ofrecer estos nuevos conocimientos en la Pontificia Universidad Javeriana, debido a que los actuales temas están enfocados en las tendencias mencionadas anteriormente.

Los startups están en crecimiento en la industria, y estas pequeñas empresas buscan un despliegue rápido y sencillo, por lo que las clásicas arquitecturas, si bien, pueden ser una opción, la tendencia es utilizar despliegues y desarrollos rápidos que brinden atributos como confiabilidad, escalabilidad, disponibilidad, entre otros, a un menor costo y esfuerzo, los cuales se pueden conseguir fácilmente con una arquitectura enfocada en microservicios.

2. Conclusiones y trabajo futuro

Se pudo entender la problemática tras los sistemas de venta de boletería en la actualidad. Tratar de modernizar un sistema que ha funcionado tanto tiempo, nos brinda la oportunidad de creer que es posible realizarlo casi que con cualquier tema. El prototipo presentado por el grupo demuestra que la utilización de herramientas tecnológicas puede aportar positivamente a casi cualquier área de negocio si son bien aplicadas.

Fue posible integrar la lectura de códigos de barras como tecnologías de identificación y captura de datos automática para facilitar la experiencia de usuario, suprimiendo la necesidad de un tiquete físico sobre el que no se tiene control.

Se cumple con el objetivo de presentar un prototipo arquitectural que ofrece una solución a la problemática de la venta de boletería, mediante la utilización de nuevas tendencias arquitecturales como la computación sin servidor o las funciones como servicio, o más general, los microservicios. Además, se utilizan herramientas que los usuarios fácilmente pueden conseguir, si es que ya no la poseen.

Para un startup, la utilización de este tipo de herramientas puede significar un ahorro económico al no tener que pensar en servidores, problemas de escalabilidad o disponibilidad, sino únicamente en el desarrollo de sus aplicaciones, mientras puedan adaptarse a este modelo. Además del rápido despliegue que un sistema desarrollado con este tipo de arquitectura y metodologías puede ofrecer.

Considerando las observaciones obtenidas en la validación por parte de expertos, la arquitectura propuesta se presenta como una aproximación coherente para atacar las problemáticas presentadas a la hora de adquirir boletería. Sin embargo, es evidente que las necesidades de las empresas organizadoras de eventos son mucho mayores a las contempladas por las funcionalidades implementadas en el prototipo final del proyecto. Por esto, es necesario considerar una arquitectura mucho más robusta que garantice la total satisfacción de las necesidades de los organizadores, es decir, que contemple las medidas de seguridad adecuadas para el manejo de datos financieros de los usuarios, distribución geográfica de los asistentes y el crecimiento de la base de clientes.

IX- REFERENCIAS

[1] C. Cruz, “Así me fue de mal con Tu Boleta y el Grupo Aval”, *Las2orillas*, 02-2016. .
[2] L. F. Botero, “Boletas que no sirven para nada”, *Boletas que no sirven para nada*, 23-2012. [En línea]. Disponible en: http://www.dinero.com/opinion/columnistas/articulo/boletas-no-sirven-para-nada/149285. [Consultado: 30-abr-2018].
[3] Agencia Afp, “Por alta reventa, postergan venta de boletas para Colombia-Perú”, *ELESPECTADOR.COM*, 27-sep-2017. [En línea]. Disponible en: https://www.elespectador.com/noticias/actualidad/por-alta-reventa-postergan-venta-de-boletas-para-colombia-peru-articulo-715284. [Consultado: 30-abr-2018].
[4] CapitalColombia, “Herramienta de Visualización de Código de Barras de Cédulas Colombianas - PDF417”. [En línea]. Disponible en: https://www.capitalcolombia.com/sec-verificar_cedulas_colombianas. [Consultado: 09-mar-2018].
[5] Casa Editorial El Tiempo, “Códigos QR en las lápidas, una manera de ser ‘eterno’”, *El Tiempo*, 27-2013. [En línea]. Disponible en: http://www.eltiempo.com/archivo/documento/CMS-12898313. [Consultado: 09-mar-2018].
[6] Logyca, “La información detrás del código de barras”, *Visión LOGYCA*, 04-feb-2016. .
[7] Registraduría Nacional del Estado Civil, “Cédula de Ciudadanía”, *Registraduria Nacional del Estado Civil*. [En línea]. Disponible en: https://wsr.registraduria.gov.co/-Cedula-de-Ciudadania,3689-.html. [Consultado: 09-mar-2018].
[8] Registraduría Nacional del Estado Civil, “Censo Electoral”, *Registraduria Nacional del Estado Civil*. [En línea]. Disponible en: https://wsr.registraduria.gov.co/-Censo-Electoral,3661-.html. [Consultado: 09-mar-2018].
[9] Registraduría Nacional del Estado Civil, “Prensa - Reclame su cédula de ciudadanía para participar en las elecciones de Congreso de la República este 11 de Marzo”, *Registraduria Nacional del Estado Civil*, 2018-2018. [En línea]. Disponible en: https://wsr.registraduria.gov.co/Reclame-su-cedula-de-ciudadania.html. [Consultado: 09-mar-2018].
[10] Network Working Group, “Internet Security Glossary”. 2007.
[11] Dointech, “Control de Acceso | Sistemas de Control de Entrada y Salida”. [En línea]. Disponible en: http://www.dointech.com.co/control-de-acceso.html. [Consultado: 30-abr-2018].
[12] University of Ireland Galway, “Automatic Identification”. [En línea]. Disponible en: http://www.nuigalway.ie/staff-sites/david_osullivan/documents/unit_10_automatic_identification.pdf.
[13] C. Buenadicha, “El análisis de escalabilidad en la identificaión y el diseño de los proyectos de desarrollo”. [En línea]. Disponible en: http://e-spacio.uned.es:8080/fedora/get/tesisuned:CiencEcoEmp-Cmbuenadicha/Documento.pdf. [Consultado: 30-abr-2018].
[14] J. Lewis y M. Fowler, “Microservices”. [En línea]. Disponible en: https://martinfowler.com/articles/microservices.html.
[15] Amazon, “Aplicaciones y capacidad de computación sin servidor”. [En línea]. Disponible en: https://aws.amazon.com/es/serverless/.

[16] D. Prothero, “Serverless Architecture”. [En línea]. Disponible en: https://www.twilio.com/docs/glossary/what-is-serverless-architecture.
[17] eticket, “eticket”. [En línea]. Disponible en: https://www.eticket.co/contenido.aspx?id=5.
[18] Tuboleta, “Tuboleta - Ayuda”. [En línea]. Disponible en: http://vive.tuboleta.com/Content/Help/Faqs_es.aspx.
[19] Ticket Shop, “Ticket Shop”. 2018.
[20] Semana, “Segundos que cuestan millones: ¿cuánto tiempo espera a que un sitio web cargue?”, 27-2016. [En línea]. Disponible en: https://www.semana.com/economia/articulo/cuanto-tiempo-espera-a-que-un-sitio-web-cargue/491552.
[21] A. Greemberg, “What is password hashing?” [En línea]. Disponible en: https://www.wired.com/2016/06/hacker-lexicon-password-hashing/.
[22] W3Counter, “Browser & Platform Market Share”. [En línea]. Disponible en: https://www.w3counter.com/globalstats.php.
[23] E. Simons, “Introducing RealWorld”, 21-abr-2017. .
[24] thinkster, *RealWorld*. Thinkster, 2018.
[25] NativeScript, “How NativeScript Works”. [En línea]. Disponible en: https://docs.nativescript.org/angular/core-concepts/technical-overview.
[26] S. Ambler, “The Agile Edge: Unified and Agile | Dr Dobb’s”, 2006. [En línea]. Disponible en: http://www.drdobbs.com/the-agile-edge-unified-and-agile/184415460. [Consultado: 13-dic-2018].