e-ISSN: 1390-5902
CEDAMAZ, Vol. 12, No. 1, pp. 68–76, enero–junio 2022
DOI: 10.54753/cedamaz.v12i1.1267

Desarrollo de una aplicación informática para la búsqueda de oferta de alquiler de inmuebles en la ciudad de Loja

Development of a computer application for the search for real estate rental offer in the city of Loja

Steeven Armijos-Bravo[1,*] and Wilman Chamba-Zaragocín[1]

[1] *Carrera de Ingeniería en Sistemas, Universidad Nacional de Loja, Loja, Ecuador*
[*] *Autor para correspondencia: smarmijosb@unl.edu.ec*

Fecha de recepción del manuscrito: 26/03/2022 Fecha de aceptación del manuscrito: 21/05/2022 Fecha de publicación: 30/06/2022

Resumen—Encontrar un bien inmueble donde vivir para satisfacer la necesidad de vivienda cuando se reside, trabaja o estudia fuera de la ciudad natal, es algo importante que requiere de tiempo y dinero. En base a ello, el presente artículo consistió en desarrollar una aplicación informática para la búsqueda de oferta de alquiler de inmuebles para los habitantes de la ciudad de Loja, el cual consta de 4 fases, en la primera fase se realizó una revisión de trabajos similares para identificar requisitos generales, luego se aplicó encuestas a 186 personas para adquirir requisitos que contrasten con los obtenidos anteriormente, dando como resultado el documento de especificación de requerimientos según el estándar IEEE830. En la segunda fase se llevó a cabo el diseño de la aplicación, donde se utilizó la metodología XP y el modelo 4+1 respectivamente para organizar la aplicación, de igual manera se utilizó la arquitectura cliente servidor multicapa para separar la lógica de presentación, lógica de negocio y la lógica de datos; en la tercera fase se codificó el web service API REST, el cliente web y móvil. Finalmente, en la cuarta fase se realizó pruebas de funcionalidad en base a los criterios de aceptación de las Historias de Usuario, las pruebas de carga y estrés obteniendo tiempos de respuesta adecuados de la petición entre el cliente y el servidor, y las pruebas de usabilidad se llevaron a cabo en un ambiente controlado donde se encuestó a 63 personas identificando que la aplicación es fácil de usar y ayuda a reducir el tiempo de búsqueda de un bien inmueble.

Palabras clave—Desarrollo de software, Aplicación web, Búsqueda de inmueble, Metodología XP, Stack MEAN

Abstract—Finding a real estate where to live to satisfy the need for housing when living, working or studying outside the hometown, is something important that requires time and money. Based on this, the present article consisted of developing a computer application for the search of real estate rental offer for the inhabitants of the city of Loja, which consists of 4 phases, in the first phase a review of similar works was carried out to identify general requirements, then surveys were applied to 186 people to acquire requirements that contrast with those obtained previously, resulting in the requirements specification document according to the IEEE830 standard. In the second phase the application design was carried out, where the XP methodology and the 4+1 model respectively were used to organize the application, in the same way the multilayer client server architecture was used to separate the presentation logic, business logic and data logic; in the third phase the web service API REST, the web and mobile client were coded. Finally, in the fourth phase, functionality tests were performed based on the acceptance criteria of the User Stories, load and stress tests obtaining adequate response times of the request between the client and the server, and usability tests were carried out in a controlled environment where 63 people were surveyed identifying that the application is easy to use and helps to reduce the search time of a real estate.

Keywords—Software development, Web application, Property search, XP methodology, MEAN stack

INTRODUCCIÓN

Las personas que migran a otras ciudades por motivos de estudio, trabajo o para mejorar su calidad de vida (Jara Escobar, 2019) se enfrentan ante la realidad de buscar una vivienda con buenas adecuaciones y acorde a su situación económica, pero muchas de las veces suelen verse afectados debido a que salen a buscar por las distintas localidades ocasionando gastos en transporte, y más importante aún, el tiempo empleado para dicho proceso de búsqueda, esto provoca que, debido a la necesidad de conseguir alojamiento obliga a estas personas a optar por alquilar viviendas con malas adecuaciones y en ubicaciones peligrosas, influyendo así, en el desarrollo integral y en las oportunidades laborales, sociales, económicas y de educación (Cevallos, 2019).

Es así que el Art. 30 de la Constitución Ecuatoriana aprobada en 2008 menciona lo siguiente: “Las personas tienen derecho a un hábitat seguro y saludable, y a una vivienda

adecuada y digna, con independencia de su situación social y económica" (Ecuador Constitución de la república del Ecuador, 2008); con ello el Estado protege el derecho de las personas que no tienen la capacidad económica de adquirir una vivienda para que al menos puedan tener acceso a una vivienda alquilada a un precio justo y razonable.
De igual manera, El Instituto Nacional de Estadística y Censos "INEC" ejecutó la Encuesta Nacional de Alquileres "ENALQUI – 2013", donde indica que la ciudad de Loja cuenta con 5.551 viviendas de las cuales 2.111 son arrendadas Inec (2013), del mismo modo el porcentaje de la población de cinco y más años tienen teléfono celular inteligente, además, el uso de internet móvil ha incrementado desde 2010 de un 2.4 % a 48.7 % (Iniguez Pineda, 2018).

Por lo mencionado anteriormente, se propone el desarrollo de una aplicación informática a la cual se denominó "LojaHouse" con el objetivo de dar respuesta a la pregunta de investigación: ¿El desarrollo de una Aplicación Informática optimizará el proceso de búsqueda de oferta de alquiler de inmuebles para los habitantes de la ciudad de Loja?

MATERIALES Y MÉTODOS

Se utilizó la metodología XP ya que resalta por contar con la mayor cantidad de información disponible y por ser la más popular (Letelier y Penadés, 2017). Las fases que se utilizaron se describen a continuación:

Fase 1. Planificación.

- Se elaboró la revisión de trabajos similares al caso de estudio para obtener requisitos generales.
- Se realizó el levantamiento de requisitos (funcionales y no funcionales), por medio de la técnica de la encuesta, los cuales fueron necesarios para establecer la frontera del sistema y la arquitectura de la misma.
- Se elaboró el documento de especificación de requisitos utilizando el estándar IEEE830 el cual indica que un buen documento de requisitos debe contemplar toda la información presentada en dicho estándar y que en el resultado de esta fase se debe producir un documento de especificación de requisitos en el que se describa lo que el futuro sistema debe hacer (Monteferrer Agut, 2001).
- Se elaboró un prototipo de la aplicación informática y se definieron las historias de usuario y sus criterios de aceptación.

Fase 2. Diseño.

- Se empleó el modelo arquitectónico 4+1 (Kruchten, 2006), mismo que describe la arquitectura del software en base a múltiples vistas, como son:
    - **vista lógica:** Representa las funcionalidades y el servicio que proporciona a los usuarios.
    - **vista de procesos:** Describe los procesos de la funcionalidad del sistema.
    - **vista de despliegue:** Describe los componentes del sistema con el fin que el programador entienda las interacciones que existen.
    - **vista física:** Muestra los componentes físicos del sistema.
    - **vista de escenarios:** Muestra la interacción que tienen los actores y los escenarios donde se va a desenvolver el sistema.

Fase 3. Codificación.

- Se desarrolló la fase de codificación de la aplicación informática.

Fase 4. Pruebas.

Se desarrolló la fase de pruebas de la Metodología XP:
- Pruebas al API REST.
- Pruebas de carga y estrés.
- Pruebas de caja negra.
- Pruebas de usabilidad.
- Pruebas de aceptación.

RESULTADOS

A. Determinar los requisitos necesarios para desarrollar la Aplicación Informática.

Se realizó una revisión de trabajos similares para poder obtener requerimientos generales donde se analizó el nivel de entendimiento y si es utilizable en la aplicación informática, obteniendo como resultado los requisitos comunes entre los 7 trabajos seleccionados (Ver Tabla 1). Para visualizar el documento detallado, acceder a: https://acortar.link/pf8TEu. De igual manera se diseñó las encuestas y se las aplicó mediante la difusión a través de contactos de WhatsApp y redes sociales como Facebook a personas que ofertan bienes inmuebles en alquiler (arrendadores) y a personas que buscan inmuebles ofertados en alquiler (arrendatarios) en un tiempo de 10 días.

Tabla 1: Requerimientos comunes entre los trabajos relacionados

Código	Requerimiento
RF001	Mostrar alquileres.
RF002	Buscar alquileres.
RF003	Gestionar publicaciones.
RF004	Contactarse con el arrendador.
RF005	Mostrar propiedades.
RF006	Registro de Usuario.
RF007	Búsqueda de Datos.
RF008	Detalle de Vivienda en Alquiler.
RF009	Llamada Telefónica al oferente del alquiler.
RF010	Envío de mensaje al oferente del alquiler.
RF011	Publicación de Alquiler.
RF012	Registrar vivienda.

Se utilizó el estándar IEEE 830 con la finalidad de poder tener un documento de especificación de requerimientos del sistema, donde se definió los requerimientos funcionales y no funcionales para un mejor entendimiento del sistema y determinar la frontera del mismo. En la Tabla 2 se muestra los requerimientos funcionales. Para ver la especificación de requerimientos acceder a: https://acortar.link/zyq0ko

e-ISSN: 1390-5902
CEDAMAZ, Vol. 12, No. 1, pp. 68–76, enero–junio 2022
DOI: 10.54753/cedamaz.v11i2.1267

Tabla 2: Requerimientos funcionales

Código	Requerimiento
RF001	Registrar Usuarios
RF002	Autenticar Usuarios.
RF003	Gestionar cuentas de usuarios.
RF004	Gestionar perfil de usuario.
RF005	Gestionar inmueble.
RF006	Gestionar solicitudes de reserva de alquiler de un bien inmueble.
RF007	Gestionar contratos de alquiler.
RF008	Alquilar bien inmueble.
RF009	Visualizar contratos de alquiler interno.
RF010	Visualizar información de un bien inmueble.
RF011	Gestionar solicitudes de reserva de alquiler de un bien inmueble.
RF012	Buscar un bien inmueble.
RF013	Cambiar contraseña.
RF014	Resetear contraseña.
RF015	Publicar un anuncio.
RF016	Enviar mensajes por WhatsApp.
RF017	Realizar una llamada telefónica.
RF018	Recibir notificación de un bien inmueble ofertado.
RF019	Seleccionar filtros de búsqueda.
RF020	Enviar mensajes al administrador.
RF021	Visualizar mensajes al administrador.
RF022	Eliminar mensajes.
RF023	Gestionar servicios básicos.

B. Diseñar e implementar la Aplicación Informática para la búsqueda de alquiler de inmuebles en base a la metodología XP

En esta sección se describe las actividades realizadas en el diseño e implementación de la Aplicación Informática en base a la metodología XP, detallando cada una de sus fases.

1. **Planificación**.
   En (López Gil, 2018) con respecto a la metodología XP, señala que, en esta primera fase, la comunicación con el cliente es esencial para poder definir los requisitos, mismos que permiten crear las historias de usuario que describen la funcionalidad del software que se va a construir.

   a. Equipos integrantes y roles

   El equipo de trabajo encargado de la implementación de la aplicación LojaHouse, se detalla en la Tabla 3.

   Se identificó tres tipos de usuarios que intervienen en la Aplicación Informática, los cuales se detallan en la Tabla 4.

   b.Historias de Usuario

   La metodología XP utiliza la técnica de Historias de Usuario (HU) para especificar los requisitos del software, describiendo brevemente las características que el sistema debe tener, ya sean requisitos funcionales como no funcionales. Las HU deben ser comprensibles y delimitadas para que el programador pueda implementarlas en el menor tiempo posible (López Gil, 2018) . Las HU fueron creadas a partir de los requerimientos funcionales definidos en el documento de especificación de

Tabla 3: Equipos integrantes y roles

Rol	Persona
Programador	Steeven Michael Armijos Bravo.
Jefe de Proyecto	Ing. Wilman Patricio Chamba Zaragocín.
Cliente	Arrendador - Arrendatario
Encargado de Pruebas	Steeven Michael Armijos Bravo.
Encargado de seguimiento	Ing. Wilman Patricio Chamba Zaragocín.
Entrenador	Ing. Wilman Patricio Chamba Zaragocín.
Gestor	Steeven Michael Armijos Bravo.

requerimientos, donde se obtuvo un total de 18 HU. Se puede acceder a las HU y su respcetiva estimación a través del siguiente enlace: https://acortar.link/Ngvic5

2. **Diseño**
   a. Vista General del Sistema En la Figura 1, se puede observar la vista general del sistema LojaHouse, donde se especifica dos tipos de cliente (Frontend): cliente web, el cual se desarrolló con el framework Angular empleando el patrón de diseño Modelo Vista Controlador (MVC), y la aplicación móvil que se utilizó el framework Flutter aplicando el patrón Lógica de Negocio de Componente (BLoC). Para la creación del servicio web API-REST (Backend), se aplicó el patrón de diseño MVC y se utilizó NodeJs con el framework ExpressJs, el mismo que se comunica con la base de datos MongoDB a través del Object Document Mapping (ODM) Mongoose, de igual forma se utilizó el servicio Cloudinary el cual permite almacenar imágenes, y finalmente se configuró el servidor web API REST para el envío de notificaciones mediante el servicio de Firebase Cloud Messaging.

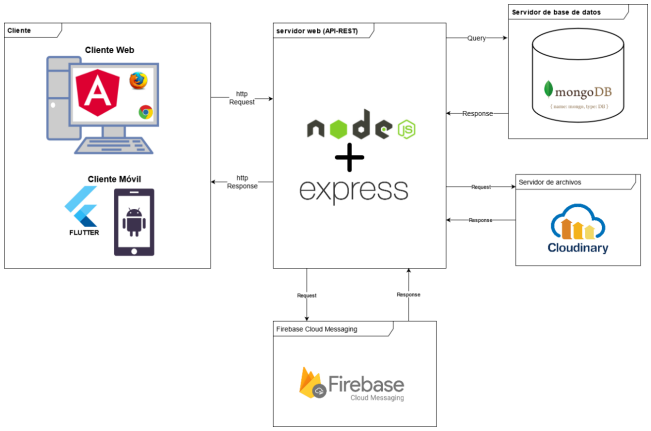


Fig. 1: Vista General del Sistema

b. Arquitectura de Software

En la Tabla 5 se puede visualizar como se encuentra definida la arquitectura de software de la aplicación informática LojaHouse utilizando el modelo arquitectónico 4+1 el mismo que propone describir la arquitectura de software por medio de diferentes vistas, como son: vista lógica, vista de despliegue, vista de escenarios,

Tabla 4: Tipo de usuario

Tipo de usuario	Descripción
Arrendador	El usuario arrendador tiene acceso a las siguientes funcionalidades: -Registrar Usuarios. -Autenticar Usuarios. -Gestionar inmueble. -Gestión de solicitudes de reservas de alquiler de un bien inmueble. -Gestión de contrato de alquiler. -Alquilar bien inmueble. -Visualizar información de un bien inmueble. -Visualizar contratos de alquiler interno. -Cambiar contraseña. -Resetear contraseña. -Publicar un anuncio. -Gestionar perfil de usuario. -Enviar mensaje al administrador.
Arrendatario	El usuario arrendatario tiene acceso a las siguientes funcionalidades: -Registrar Usuarios. -Autenticar Usuarios. -Visualizar información de un bien inmueble. -Buscar un bien inmueble. -Gestión de solicitudes de reservas de alquiler de un bien inmueble. -Gestionar contratos de alquiler. -Visualizar contratos de alquiler interno. -Cambiar contraseña. -Resetear contraseña. -Enviar mensajes por WhatsApp. -Realizar una llamada telefónica. -Recibir notificación de un bien inmueble ofertado. -Seleccionar filtros de búsqueda. -Gestionar perfil de usuario. -Enviar mensaje al administrador.
Administrador	El usuario administrador tiene acceso a las siguientes funcionalidades: -Registrar Usuarios. -Autenticar Usuarios. -Gestionar cuentas de usuarios. -Gestionar perfil de usuario. -Visualizar mensajes al administrador. -Eliminar mensajes. -Gestión de servicios básicos.

vista física y vista de procesos (Kruchten, 2006).

3. **Codificación**

Se codificó la aplicación LojaHouse, utilizando un stack de tecnologías basadas en el lenguaje de programación JavaScript denominado Stack MEAN (MongoDB, Express, Angular, Node) y para la parte móvil se utilizó el framework Flutter y el lenguaje de programación Dart. Enn la Figura 2 se muestra la estructura del servicio web API REST, de igual manera en la Figura 3 se muestra el patrón MVC utilizado.

La ruta se encarga de reenviar las solicitudes admitidas hacia el controlador apropiado. La Figura 4 muestra

Tabla 5: Arquitectura 4+1

Vista	Elemento Modelado
Vista de Escenarios	Diagrama de casos de uso.
Vista Lógica	Modelo Conceptual, Diagrama de Clases.
Vista Física	Diagrama de Despliegue.
Vista de Despliegue	Diagrama de Componenetes.
Vista de Procesos	Diagrama de Actividad.

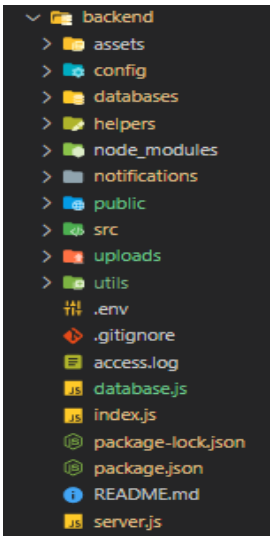


Fig. 2: Estructura del Servicio Web API-REST

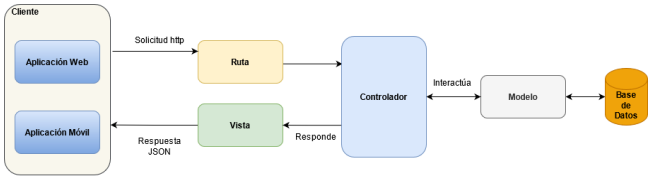


Fig. 3: Patrón de diseño Modelo-Vista-Controlador

la definición de las rutas para el módulo de visitas, las mismas que redirigen la petición hacia el controlador específico.

```
const express = require('express');
var mdwVerificarToken = require('../middlewares/autenticacion');
const router = express();

const {
} = require('../controllers/visita.controllers')

//SE OBTIENE SOLO LAS VISITAS A LOS INMUEBLES DEL ARRENDADOR
router.get('/obtenervisitas/:desde', mdwVerificarToken.verificaToken, obtenerVisitas);
//SE OBTIENE SOLO LAS VISITAS A LOS INMUEBLES DEL ARRENDADOR, esto es para el dashboard
router.get('/obtenervisitas-contador', mdwVerificarToken.verificaToken, obtenerSolicitudVisitasPendientes);

//SE OBTIENE SOLO LAS VISITAS A LOS INMUEBLES DEL ARRENDADOR-ADMINISTRADOR
router.get('/administrador/arrendador/visitas/:desde', mdwVerificarToken.verificaToken, obtenerVisitasArrendatarioAdministrador);

//CRUD ARRENDADOR (VISITA)
router.post('/crearvisita', mdwVerificarToken.verificaToken, crearVisita);
router.put('/actualizarvisita/:id', mdwVerificarToken.verificaToken, actualizarVisita);
router.put('/eliminarvisita/:id', mdwVerificarToken.verificaToken, eliminarVisita);

//RUTAS DEL USUARIO ARRENDADOR PARA ACEPTAR LAS VISITAS
router.put('/arrendador/aceptarvisita/:id', aceptarVisita);

//RUTA PARA OBTENER TODAS LAS VISITAS QUE EL USUARIO ARRENDATARIO HA SOLICITADO
router.get('/arrendatario/visitasolicitada/:desde', mdwVerificarToken.verificaToken, obtenerVisitasSolicitadas);
router.get('/arrendatario/visitasolicitadamovil1', mdwVerificarToken.verificaToken, obtenerVisitasSolicitadasMovil1);
//RUTAS DEL USUARIO ARRENDATARIO PARA LAS VISITAS
router.get('/arrendatario/obtenervisita/:id', mdwVerificarToken.verificaToken, obtenerVisitaEspecificaArrendatario);
router.get('/arrendador/obtenervisita/:id', mdwVerificarToken.verificaToken, obtenerVisitaEspecificaArrendador);

module.exports = router;
```

Fig. 4: Ruta para el módulo visita

El modelo contiene la representación de los datos del dominio de la aplicación, es decir, las clases. La Figura 5 muestra la definición del modelo para el módulo de visitas utilizando el ODM Mongoose, el cual permite

e-ISSN: 1390-5902
CEDAMAZ, Vol. 12, No. 1, pp. 68–76, enero–junio 2022
DOI: 10.54753/cedamaz.v11i2.1267

crear el esquema de la base de datos, además permite realizar validaciones y crear consultas.

```
let mongoose = require('mongoose');

let Schema = mongoose.Schema;


let estadosValidos = {
    values: ['ACEPTADA', 'RECHAZADA', 'PENDIENTE', 'ATENDIDA', 'ELIMINADA'],
    message: '{VALUE} no es un estado permitido'
}

let visitaSchema = new Schema({
    fecha: {type: Date, required: [true, 'La fecha es necesaria']},
    descripcion: {type: String, required: [true, 'La descripción es necesaria']},
    estado: {type: String,required:false, default: 'PENDIENTE', enum: estadosValidos},
    inmueble: {type: Schema.Types.ObjectId, ref: 'Inmueble'},
    usuarioarrendatario: {type: Schema.Types.ObjectId, ref: 'Usuario'},
}, {timestamps: true, versionKey: false}, {collection: 'visitas'});

module.exports = mongoose.model('Visita', visitaSchema);
```

Fig. 5: Modelo para el módulo visita

El controlador es el que interactúa con el modelo para obtener los datos solicitados y representarlos a través de la vista. En la Figura 6 se muestra el código del controlador de aceptar visita, aquí es donde se procesan los datos de la solicitud y recuperará la información necesaria del modelo para luego entregárselos a la vista para que los procese el cliente.

```
aceptarVisita: async (req, res) => {
  let id = req.params.id;
  const { estado } = req.body;
  await visitaModel.findById(id, (err, visita) => {
    if (err) {
      return res.status(500).json({
        ok: false,
        mensaje: "Error al buscar visita",
        errors: err,
      });
    }
    if (!visita) {
      return res.status(400).json({
        ok: false,
        mensaje: "El visita con el id: " + id + " no existe",
        errors: { message: "No existe un visita con ese ID" },
      });
    }
    visita.estado = estado;
    visita.save((err, visitaGuardado) => {
      if (err) {
        return res.status(400).json({
          ok: false,
          mensaje: "Error al aceptar la visita",
          errors: err,
        });
      }
      res.status(200).json({
        ok: true,
        visita: visitaGuardado,
        mensaje: `La visita está: ${estado}`,
      });
    });
  });
```

Fig. 6: Controlador para aceptar una visita

La vista es utilizada por el controlador para representar los datos, en este caso la respuesta en formato JSON. En la Figura 7 se muestra la respuesta que se envía desde el controlador en formato JSON para que el cliente pueda procesarla.

Diseño Final de interfaz de usuario de la aplicación LojaHouse. En la Figura 8 se muestra la interfaz de

```
    if (err) {
      return res.status(400).json({
        ok: false,
        mensaje: "Error al aceptar la visita",
        errors: err,
      });
    }
    res.status(200).json({
      ok: true,
      visita: visitaGuardado,
      mensaje: `La visita está: ${estado}`,
    });
```

Fig. 7: Respuesta en formato JSON para aceptar una visita

usuario de la página principal de la aplicación LojaHouse.

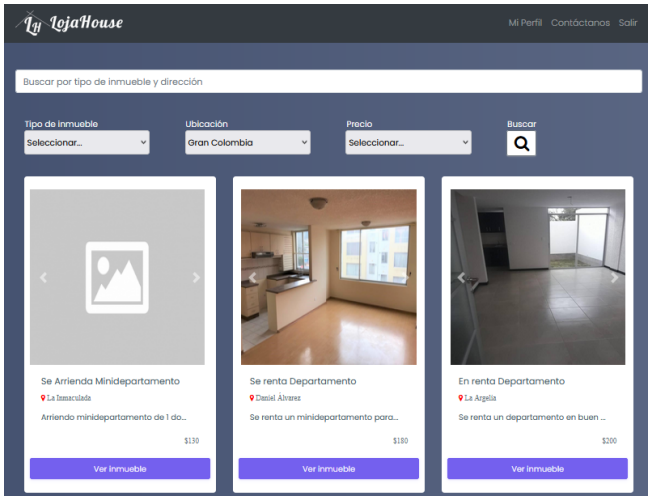


Fig. 8: Página principal

El Proyecto LojaHouse se encuentra alojado en el repositorio de control de versiones GitHub:
Backend: `https://acortar.link/DnPbOR`
Frontend: `https://acortar.link/DK0rbY`
Aplicación móvil: `https://acortar.link/TleRts`

C. Evaluar la funcionalidad y usabilidad de la Aplicación Informática en un ambiente controlado o simulado.

4. Pruebas

En esta sección se describe la cuarta fase de la metodología XP, la fase de pruebas del sistema para evaluar la funcionalidad e identificar errores en el software.

Pruebas al Servicio Web API REST

Para las pruebas del API REST se utilizó la herramienta Postman, la cual permitió el envío de peticiones http (get, put, delete, post) hacia el servicio web API REST, obteniendo una respuesta en formato json, el mismo que permitió verificar el correcto funcionamiento del Backend e identificar el tiempo que demora la respuesta en cada petición realizada.

En la Tabla 6 se muestra los test realizados donde se utilizó la herramienta Postman considerando los

siguientes parámetros: url, token, método, resultado y tiempo de respuesta de la petición, para ello se realizó el test a las peticiones principales del API REST.

Tabla 6: Pruebas unitarias servicio web API REST - postman

URL	Método	Resultado	T.respuesta
localhost:3000/ usuario/crearusuario	POST	Registra la cuenta del usuario	2160 ms.
http://localhost: 3000/ login	POST	Inicia sesión	238 ms.
localhost: 3000/ usuario/ reseteopassword	PUT	Resetea el password	1860 ms.
localhost:3000/ mensaje/ crearmensaje	POST	Envia un mensaje	273 ms.
localhost:3000/ busqueda/ coleccion/ inmuebles/ CASA/ Gran Colombia/ 150-200	GET	Permite filtrar bienes inmuebles	297 ms.
localhost:3000/ inmueble/ obtenerinmueble/ publico/ 60a88386fdbc902cfc00fc22	GET	Visualiza información del inmueble	241 ms.
localhost:3000/ inmueble/ crearinmueble	POST	Crea un bien inmueble	244 ms.
localhost:3000/ inmueble/ desactivarinmueble/ 60ed2b0f064bcc3ba8f21378	PUT	Publica un bien inmueble	578 ms.
localhost:3000/ visita/ crearvisita	POST	Permite generar una solicitud de alquiler	250 ms.
localhost:3000/ inmueble/ desactivarinmueble/ 60ed2b0f064bcc3ba8f21378	PUT	Acepta una visita de alquiler	239 ms.
localhost:3000/ contrato/ crearcontrato	POST	Genera un contrato de alquiler	357 ms.
localhost:3000/ contrato/ acuerdo/ 60ed32a0db551b3bbcd02525/ aceptar	POST	Acepta un contrato de alquiler	411 ms.

Pruebas de caja negra

Las pruebas de caja negra se llevaron a cabo en base a los criterios de aceptación de las Historias de Usuario para determinar si el software cumple o no con los requisitos de software. La Tabla 7 muestra el ejemplo para el caso de prueba de caja negra para la creación de un inmueble.

Tabla 7: Caso de prueba de caja negra - crear inmueble

Criterio de aceptación	Cumple
Mostrar el formulario de registro de un nuevo inmueble a través del botón Nuevo Inmueble de la pantalla gestión de inmuebles.	Si
Cargar las opciones de tipo de inmueble como: casa, departamento, cuarto, minidepartamento en la pantalla de registro de nuevos inmuebles.	Si
Seleccionarlos servicios que incluye un bien inmueble.	Si
Validar campos obligatorios.	Si
Mostrar un mensaje indicando que hay campos obligatorios vacíos.	Si
Controlar si el bien inmueble posee garantía, si no se ingresa un monto, por defecto será 0.	Si
Guardar el inmueble al dar clic en el botón “Guardar inmueble”.	Si
Registrar el bien inmueble y mostrar un mensaje notificando al usuario que se ha registrado su inmueble.	Si
Redirigir a otra pantalla donde se podrá subir las imágenes del inmueble.	Si
Cargar imágenes al bien inmueble.	Si
Quitar imágenes (de ser necesario) del bien inmueble.	Si
Mostrar un mensaje indicando lo siguiente: “Imágenes cargadas exitosamente”.	Si

Pruebas de carga y estrés

El backend de la aplicación LojaHouse se encuentra alojado en los servidores de Heroku, y el frontend en los servidores de Firebase Hosting, de google. Para conocer el nivel de carga que puede soportar la aplicación y su rendimiento, se realizó pruebas de carga y estrés.

En la Figura 9 se muestra la configuración del grupo de hilos, donde se define las peticiones de tipo GET y POST que se estima podrían producir cuellos de botella, para ello se configuró 1000 peticiones para un periodo de un segundo que se ejecutarán de manera concurrente.

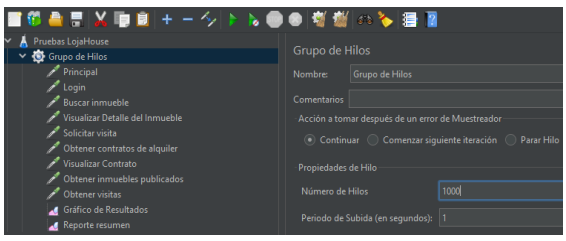
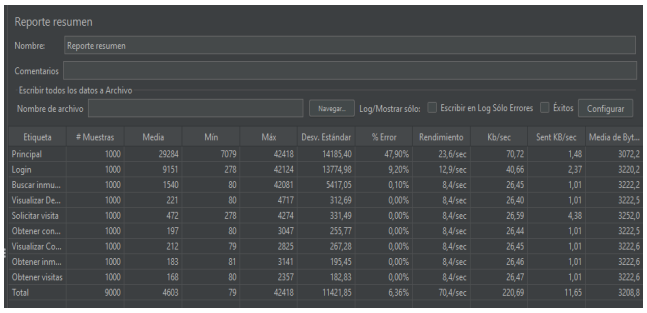


Fig. 9: Configuración de peticiones

e-ISSN: 1390-5902
CEDAMAZ, Vol. 12, No. 1, pp. 68–76, enero–junio 2022
DOI: 10.54753/cedamaz.v11i2.1267

En la Figura 10 se puede observar el resultado en forma de resumen, lo cual permite conocer el error obtenido en la prueba, en este caso se obtuvo un porcentaje de 6.36 %, es decir, la aplicación LojaHouse tiende a fallar más al realizar peticiones concurrentes en la página principal, por otro lado, la página de login tiende a fallar en menor medida. Aun así, se puede evidenciar que la aplicación soportó 1000 peticiones concurrentes presentando un rendimiento estable, dando un total de 70,4 % de rendimiento, por lo que se concluye que la aplicación funciona de forma correcta con tiempos de respuesta mínimos de 79 ms y como máximo de 42.418 ms.


Reporte resumen

Nombre: Reporte resumen

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Mostrar sólo: | Escribir en Log Sólo Errores | Éxitos | Configurar

Etiqueta	# Muestras	Media	Min	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Byt...
Principal	1000	29284	7079	42418	14185,40	47,90%	23,6/sec	70,72	1,48	3072,2
Login	1000	9151	278	42124	13774,98	9,20%	12,9/sec	40,66	2,37	3220,2
Buscar inmu...	1000	1540	80	42081	5417,05	0,10%	8,4/sec	26,45	1,01	3222,2
Visualizar De...	1000	221	80	4717	312,69	0,00%	8,4/sec	26,40	1,01	3222,5
Solicitar visita	1000	472	278	4274	331,49	0,00%	8,4/sec	26,59	4,38	3252,0
Obtener con...	1000	197	80	3047	255,77	0,00%	8,4/sec	26,44	1,01	3222,5
Visualizar Co...	1000	212	79	2825	267,28	0,00%	8,4/sec	26,45	1,01	3222,6
Obtener inm...	1000	183	81	3141	195,45	0,00%	8,4/sec	26,46	1,01	3222,6
Obtener visitas	1000	168	80	2357	182,83	0,00%	8,4/sec	26,47	1,01	3222,6
Total	9000	4603	79	42418	11421,85	6,36%	70,4/sec	220,69	11,65	3208,8

Fig. 10: Reporte de resumen de resultados de la ejecución de la prueba de carga y estrés

Pruebas de Usabilidad

Para las pruebas de usabilidad se empleó el método del experimento y el método empírico de la encuesta mediante un cuestionario estandarizado PSSUQ (Cuestionario de Usabilidad del Sistema Posterior al Estudio) utilizado en (Tobar Ibarra, 2018). Este cuestionario permite obtener respuestas en base a la escala de Likert la cual se aplicó luego que el usuario interactuó con la Aplicación Informática LojaHouse.

Para llevar a cabo las pruebas se procedió a invitar de manera voluntaria a los participantes en rol de arrendadores por medio de las Aplicaciones Facebook y WhatsApp; para ello se envió la dirección web (https://frontendlh.web.app//login) desde el cual accedían a la aplicación LojaHouse, para el caso de los participantes de la carrera de Ingeniería en Sistemas/Computación, en su rol de arrendatarios, se solicitó descargar la aplicación móvil de la Play Store de Google para llevar a cabo las pruebas, las mismas que fueron efectuadas en el lapso de 27 días obteniendo un total de 63 personas encuestadas que probaron la aplicación LojaHouse donde se obtuvo como resultado que la aplicación es aceptada por los usuarios, es fácil de usar y sobretodo, permite agilizar la búsqueda de oferta de alquiler de un bien inmueble, donde normalmente los usuarios emplean 1 día como mínimo, hasta más de 3 días, por lo cual, mediante las pruebas realizadas dicha actividad se demora de 10 a 30 minutos como mínimo, hasta máximo 1 día.

DISCUSIÓN

La revisión de los trabajos (Ríos Pinzón, 2008), (Ruano Arranz, 2016), (Jara Castillo, 2015), (Herrera Arizaga, 2015), (Lázaro Hernández, 2018), (Loor Villamar y Morocho Ramos, 2016), (Herrera Nole, 2016) ayudó a identificar 68 requisitos de los cuales 12 se tomaron en cuenta, para ello se realizó una evaluación bajo el criterio del autor con el objetivo de determinar el nivel de entendimiento y su utilidad. Este trabajo planteó el uso de la entrevista para la obtención de requisitos, pero a consecuencia de las medidas restrictivas que surgieron por motivos de la crisis sanitaria a nivel mundial, COVID-19, se utilizó la técnica de la encuesta en un formato digital haciendo uso de la herramienta Google Forms, las cuales fueron difundidas a través de las Aplicaciones de Facebook y WhatsApp logrando llegar a las personas que residen en la ciudad de Loja. Los resultados obtenidos permitieron corroborar con los requisitos de los trabajos similares, además, los requisitos propuestos se diferencian en cuanto a realizar una llamada telefónica al arrendador, el envío de mensajes por WhatsApp, la solicitud de visitas de alquiler, el envío de mensajes al administrador, generación de contrato de alquiler interno, filtrar por el tipo de inmueble, la dirección y el precio, recibir notificaciones en la aplicación; de igual manera, como requisito no funcional se consideró desarrollar la aplicación web y móvil. Como resultado final se obtuvo el documento de Especificación de Requerimientos de Software en base al estándar IEEE-830, mismo que ha sido objeto de estudio en los trabajos revisados anteriormente, por lo que se identificó un total de 23 requisitos funcionales y 9 requisitos no funcionales, además la tecnología a utilizar, a diferencia de los trabajos revisados que utilizaban varios lenguajes de programación para el Backend (php, python) y para el Frontend (HTML, CSS, JavaScript), por lo que para este estudio se definió un stack de tecnologías basados en el lenguaje de programación JavaScript y para la aplicación móvil el lenguaje de programación Dart.

Las fases de la metodología XP permitieron desarrollar el proyecto de manera exitosa. En primera instancia se logró definir las HU en base al documento de requerimientos, obteniendo así un conjunto de 18 HU que a su vez se realizó una estimación de tiempo de implementación general, ya que una metodología no es restrictiva, es por ello que se optó por realizar una estimación completa inicial. Para reforzar el diseño del sistema, se utilizó el modelo arquitectónico 4+1 el cual permitió modelar y documentar el diseño de la aplicación informática para un mejor entendimiento en base a 5 vistas bien definidas. A partir de ello, en la fase de Codificación de la metodología XP, se facilitó en gran medida, ya que se eligió los módulos de forma alternativa para llevar a cabo la implementación de los mismos. Cabe recalcar que existieron funcionalidades que generaron retrasos en la implementación, por ejemplo el manejo de imágenes, envío de notificaciones push, envío de correos electrónicos, entre otras. Al elegir el Framework Flutter con el lenguaje de programación Dart, la curva de aprendizaje fue fácil con un grado de dificultad por falta de conocimientos en cuanto al emplear el patrón bloc para el manejo de estados, pero permitió llevar a cabo el desarrollo de la aplicación móvil de manera exitosa ya que se puede crear aplicaciones elegantes que se ven igual tanto en IOS como en Android, además es una tecnología que está apoderándose de la programación web y de escritorio, cabe recalcar que la aplicación móvil se ha generado específicamente para Android porque no se dispone del hardware necesario para realizar pruebas en dispositivos

IOS. Por otro lado, la aplicación LojaHouse permite agregar futuras funcionalidades ya que se compone de una arquitectura cliente servidor multicapa, la cual permite que tanto la aplicación web y móvil se comuniquen a través del Servicio Web API REST.

La evaluación de la aplicación LojaHouse se llevó a cabo mediante diversas pruebas que permitieron garantizar que el software funciona correctamente y cumple con las especificaciones definidas en el documento de Especificación de Requerimientos. Se realizaron casos de prueba de caja negra en base a las Historias de Usuario lo que permitió encontrar algunas fallas en la funcionalidad de la aplicación, las cuales se lograron solventar de manera exitosa y cumplir con los requerimientos especificados. Por otro lado, las pruebas de carga y estrés permitieron detectar en qué páginas tiende a fallar la aplicación, por lo que la página principal y la de iniciar sesión fueron las que presentan cuellos de botella al tener más de 1000 usuarios concurrentes, en este caso influye el servicio de hosting que se utilizó, al tener una versión gratuita en los servidores de Heroku y Firebase, el tiempo de respuesta tiende a ser más lento. Finalmente las pruebas de usabilidad realizadas en base a un cuestionario estandarizado PSSUQ (Cuestionario de Usabilidad del Sistema Posterior al Estudio) utilizado en (Tobar Ibarra, 2018), permitió establecer que la aplicación es aceptada por los usuarios, es fácil de usar y sobretodo, permite agilizar la búsqueda de oferta de alquiler de un bien inmueble, donde normalmente los usuarios emplean de 1 día como mínimo, hasta más de 3 días, por lo cual, mediante las pruebas realizadas dicha actividad se demora de 10 a 30 minutos como mínimo, hasta máximo 1 día.
De forma general, la aplicación facilita la búsqueda y control de los bienes inmuebles, de los contratos de alquiler interno y las visitas, por lo que se obtuvo como resultado que la aplicación es fácil de usar y es aceptada por los usuarios.

CONCLUSIONES

El desarrollo de la aplicación LojaHouse permite optimizar la búsqueda de bienes inmuebles a los usuarios que desean alquilar una vivienda en la ciudad de Loja, con respecto al tiempo de búsqueda, donde normalmente empleaban de 1 día como mínimo, hasta más de 3 días, por lo cual, mediante las pruebas realizadas dicha actividad se demora de 10 a 30 minutos como mínimo, hasta máximo 1 día, y como ahorra el tiempo de búsqueda también ayuda económicamente ya que el arrendatario no se dirige al bien inmueble hasta que exista una previa comunicación entre ambas partes, por otro lado, se determinó que la aplicación es aceptada por los usuarios, y que la mayoría está muy de acuerdo en que la aplicación es fácil de usar.

La revisión de trabajos similares ayudó en gran medida en la recopilación de requisitos generales, y con la encuesta se logró identificar y detallar 23 requisitos funcionales y 9 requisitos no funcionales que la aplicación debe contemplar, quedando establecidos en el documento de Especificación de Requisitos en base al estándar IEEE 830.

El desarrollo de la aplicación siguiendo la metodología XP ayuda en gran medida a organizar las actividades de forma general para el cumplimiento de las 18 historias de usuario a través de cada una de sus fases, como son: la fase de planificación, diseño, codificación y pruebas para un desarrollo exitoso del proyecto, de igual manera se utilizó el modelo arquitectónico 4+1 para tener una mejor comprensión y comunicación en el desarrollo del sistema.

La fase de pruebas ayudó a validar que la aplicación cumple con los requisitos especificados, que la aplicación es fácil de usar y sobretodo la aceptación del sistema por parte de los usuarios. Además permitieron corroborar que el rendimiento de la aplicación tenga tiempos de respuesta aceptables.

Utilizar tecnologías basadas en JavaScript como el Stack MEAN (MongoDB, Express, Angular, Node) ayudado con el Framework Flutter, permitió llevar a cabo el desarrollo de la aplicación de manera exitosa, logrando obtener un producto de software funcional que cumple con los requisitos especificados, además, reduce el tiempo de desarrollo y la curva de aprendizaje

CONTRIBUCIONES DE LOS AUTORES

"Conceptualización, SAB; metodología, SAB y WCHZ; análisis formal, SAB; investigación, SAB; recursos, SAB; curación de datos, SAB; redacción — preparación del borrador, SAB; original, SAB; redacción — revisión y edición, SAB y WCHZ; visualización, WCHZ; supervisión, WCHZ; administración de proyecto: WCHZ, adquisición de financiamiento para la investigación: SAB. Todos los autores han leído y aceptado la versión publicada del manuscrito.
Steeven Armijos-Bravo: SAB. Wilman Chamba-Zaragocín: WCHZ".

FINANCIAMIENTOS

El presente estudio fue financiado por el autor del mismo.

REFERENCIAS

Cevallos, C. (2019). *Programa de arrendamiento de vivienda social: Alternativa para reducir el déficit de vivienda social, generar mayor asequibilidad y disminuir la segregación espacial en el Distrito Metropolitano de Quito (DMQ)* (Tesis Doctoral). Descargado de http://repositorio.puce.edu.ec/bitstream/handle/22000/16312/Disertaci{ó}nCarolinaCevallosFeb-2019.pdf?sequence=1&isAllowed=y

Ecuador Constitución de la república del Ecuador, C. (2008). *artículo 30.* Descargado de https://www.oas.org/juridico/pdfs/mesicic4_ecu_const.pdf

Herrera Arizaga, M. B. (2015). *desarrollo de una aplicación web y una aplicación móvil para la gestión de alquiler y venta en una inmobiliaria* (Tesis Doctoral). Descargado de http://repositorio.utmachala.edu.ec/bitstream/48000/5125/1/TTUAIC_2015_ISIST_CD0034.pdf

Herrera Nole, J. F. (2016). *implementación de un sistema web y una aplicación móvil para administrar los servicios y actividades de una empresa inmobiliaria.* (Tesis Doctoral). Descargado de http://repositorio.utmachala.edu.ec/handle/48000/7660

Inec. (2013). *encuesta nacional de alquileres enalqui 2013* (Inf. Téc.).

e-ISSN: 1390-5902
CEDAMAZ, Vol. 12, No. 1, pp. 68–76, enero–junio 2022
DOI: 10.54753/cedamaz.v11i2.1267

Iniguez Pineda, C. F. (2018). *análisis comparativo del servicio de streaming de video de youtube entre las operadoras de datos móviles 4g en el ecuador, para proponer parámetros mínimos de calidad* (Tesis Doctoral no publicada).

Jara Castillo, J. A. (2015). *desarrollo de una aplicación web y una aplicación móvil para registrar el alquiler y ventas de una inmobiliaria utilizando scrum* (Tesis Doctoral no publicada).

Jara Escobar, J. M. (2019). *migración estudiantil. un análisis exploratorio sobre el impacto en la adaptación de los estudiantes de la carrera de licenciatura en gestión y desarrollo turístico de la universidad estatal península de santa elena, ecuador* (Tesis Doctoral no publicada).

Kruchten, P. (2006). Planos Arquitectónicos : El Modelo de " 4 + 1 " Vistas de la La Arquitectura del Software. , *12*(6), 1–16.

Lázaro Hernández, C. (2018). *diseño y desarrollo de una aplicación web para compartir piso* (Tesis Doctoral). Descargado de `http://openaccess.uoc.edu/webapps/o2/handle/10609/72425`

Letelier, P., y Penadés, C. (2017). metodologías ágiles para el desarrollo de software: extreme programming (xp). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*, 17.

Loor Villamar, A. D., y Morocho Ramos, M. A. (2016). *desarrollo e implementación de un aplicativo para dispositivos móviles con sistema operativo android y geolocalización que permita ofrecer y solicitar servicios de alquiler de casas con ubicación exacta y a la vez sugiera las más cercanas a la ubicación a* (Tesis Doctoral no publicada).

López Gil, A. (2018). *Estudio comparativo de metodologías tradicionales y ágiles para proyectos de desarrollo de software* (Tesis Doctoral). Descargado de `http://uvadoc.uva.es/handle/10324/32875`

Monteferrer Agut, R. (2001). Especificación de Requisitos Software según el estándar de IEEE 830.

Ríos Pinzón, E. G. (2008). *Desarrollo de un sistema informático para los procesos de cosecha y post-cosecha de la camaronera "pampas de Cayanca"* (Tesis Doctoral no publicada).

Ruano Arranz, M. (2016). *estupiso segovia: buscador de viviendas en alquiler en segovia para estudiantes* (Tesis Doctoral no publicada).

Tobar Ibarra, O. D. (2018). Evaluación de usabilidad de plataforma educativa acceso multidipositivos.