UNIVERSIDAD
NACIONAL
DE LA PLATA

Proyecto de Software

Informe Final

Paula Vaccaro - Facundo Miglierini

Contents

1 Abstract

En este informe procederemos a explicar todo lo referente al proceso de desarrollo del sitio web para el Club Deportivo de Villa Elisa, implementado como trabajo integrador grupal correspondiente a la materia Proyecto de Software de la Facultad de Informática de la Universidad Nacional de La Plata (UNLP). La creación de dicho programa fue posible gracias a la aplicación de los conocimientos adquiridos en la cursada, ya que hemos implementado diversas técnicas tales como la utilización de librerías (ya sea built-ins, como aquellas de libre uso almacenadas en el repositorio de GitHub[1]); conceptos relacionados a la gestión de bases de datos, la accesibilidad, la responsividad, el consumo de las APIs y el manejo de sesiones; la gestión para la autenticación de los usuarios y el uso del session storage para almacenar información; entre otros conceptos y mecanismos aprendidos durante el transcurso del semestre.

2 Palabras clave

Aplicación, Accesibilidad web, Progressive Web Apps, Unit tests, Privacidad

3 Introducción

Este trabajo condensa los resultados de un proyecto de software llevado adelante durante el primer semestre del año facultativo 2022. En base al pedido del Club Deportivo de Villa Elisa, hemos desarrollado un sitio web que permita gestionar y brindar información a las personas asociadas del club. Este sitio se divide en dos partes que mencionaremos a continuación.

Por un lado, desarrollamos una aplicación privada que provee la gestión de las personas, los pagos y las disciplinas presentes en el club, junto con una serie de opciones de configuración (sea la cantidad de elementos por página o el porcentaje de recargo para cuotas adeudadas) y la administración de los roles (permisos) para cada usuario.

Del mismo modo, elaboramos una aplicación pública, la cual permite visualizar la información de interés para la comunidad y las personas asociadas al club. A través de la misma es posible obtener información para poder contactarse con el club, conocer las disciplinas que se practican junto con sus precios y horarios, observar un carnet de socio (el cual plasma los datos personales y permite identificarse de manera rápida y sencilla a través de un código QR), consultar el estado de socio y registrar la realización de un pago de cuota.

Para construir estos aspectos, en la aplicación privada utilizamos Flask como framework y PostgreSQL como motor de base de datos (recurriendo a SQLAlchemy como ORM [por sus siglas en inglés, Object Relational Mappings], el cual nos permitió tener una capa de abstracción con la base de datos), mientras que en la aplicación pública aplicamos VueJS3 como framework. Adicionalmente, acudimos a HTML5, CSS3 y Bootstrap en la extensión total del trabajo para producir las vistas, asegurándonos de que todas sean web-responsive y estén validadas por la W3C (por sus siglas en inglés, World Wide Web Consortium: https://www.w3.org/).

4 Investigación sobre PWA y tests

Como parte de la extensión del trabajo final, precisamos investigar e implementar una PWA y distintos tests de unidad.

4.1 PWA

Las PWA (por sus siglas en inglés, Progressive Web Apps) son aplicaciones web creadas y mejoradas con APIs modernas para proporcionar a los usuarios una experiencia más confiable, dado que utilizan nuevas capacidades para proporcionar una experiencia más integrada y pueden instalarse. Consisten principalmente de un manifiesto de aplicación web, un trabajador de servicio y un script para registrar el trabajador de servicio en el explorador.

[1] Página oficial de GitHub: https://github.com

- Manifiesto de aplicación web: proporciona información sobre una aplicación web en un archivo de texto JSON, se requiere para que la aplicación web pueda descargarse y presentarse al usuario de manera similar a una aplicación nativa.
- Trabajador de servicio (Service Worker): esencialmente actúa como servidor proxy. Toma la forma de un archivo JavaScript que puede controlar el sitio web con el que está asociado, interceptando y modificando solicitudes de navegación y recursos. Cuando una aplicación solicita un recurso cubierto por el ámbito del trabajador de servicio, incluso cuando un usuario está sin conexión, el service worker intercepta la solicitud para luego decidir cómo resolverla (a través de una API, de la red o de un algoritmo local).

Para que nuestro trabajo pueda convertirse en una aplicación web progresiva, decidimos incorporar la librería vite-plugin-pwa (https://vite-pwa-org.netlify.app/), la cual nos abstrae casi completamente de la configuración necesaria para el correcto funcionamiento de una PWA.

4.2 Tests de unidad

Los tests de unidad son una manera de poner a prueba una unidad de nuestra aplicación (la pieza más pequeña de código que puede ser lógicamente aislada en un sistema). En la mayoría de los lenguajes de programación, se trata de una función, una subrutina, un método o una propiedad. Este tipo de tests ofrece las siguientes ventajas:

- Demuestran que la lógica del código está en buen estado y que funcionará en todos los casos.
- Aumentan la legibilidad del código y ayudan a los desarrolladores a entender el código base, lo que facilita hacer cambios más rápidamente.
- Sirven como documentación del proyecto.
- Se ejecutan en pocos milisegundos.
- Permiten al desarrollador refactorizar el código más adelante y tener la garantía de que la unidad sigue funcionando correctamente.
- La calidad final del código mejorará ya que, al estar realizando pruebas de manera continua, al finalizar el código será limpio y de calidad.
- Como las pruebas unitarias dividen el código en pequeños fragmentos, es posible probar distintas partes del proyecto sin tener que esperar a que otras estén completadas.

Para llevar a cabo buenas pruebas unitarias, deben estar estructuradas siguiendo las tres A's del Unit Testing. Se trata de un concepto fundamental respecto a este tipo de pruebas, que describe un proceso compuesto de tres pasos.

- Arrange (organizar): es el primer paso de las pruebas unitarias. En esta parte se definen los requisitos que debe cumplir el código.
- Act (actuar): es el paso intermedio de las pruebas, el momento de ejecutar el test que dará lugar a los resultados que deberás analizar.
- Assert (afirmar): en el último paso, es el momento de comprobar si los resultados obtenidos son los que se esperaban. Si es así, se valida y se sigue adelante. Si no, se corrige el error hasta que desaparezca.

Con respecto a nuestro trabajo, hemos implementado los test de unidad haciendo uso de la librería "Pytest", mediante la cual pusimos a prueba diversas funciones encargadas de realizar distintas acciones sobre los modelos que componen nuestra base de datos. Entre ellas, podemos mencionar la carga, borrado y modificación de la información de nuevos usuarios, socios y disciplinas.

Además, hemos puesto a prueba las APIs que permiten conectar la app privada con la pública haciendo uso de "Postman" (https://www.postman.com/), una plataforma destinada a la construcción y el uso de APIs. Mediante la misma pusimos a prueba las distintas APIs de la aplicación gestionándolas por etapas, comenzando por aquellas que no requieren el inicio de una sesión, continuando con las que funcionan con una sesión iniciada, y finalizando se pone a prueba la API encargada del cierre de sesión.

5 Análisis de accesibilidad

En Argentina, el artículo 3 de la ley N°26653 define que "se entiende por accesibilidad a los efectos de esta ley a la posibilidad de que la información de la página Web, puede ser comprendida y consultada por personas con discapacidad y por usuarios que posean diversas configuraciones en su equipamiento o en sus programas." Esta ley, que sigue los estándares internacionales formados por la W3C, define que los aspectos generales de accesibilidad web son, entre otros:

- Título de la página: la página web debe tener un título que describa su temática o propósito.
- Texto alternativo para imágenes: las alternativas textuales para las imágenes están orientadas a aquellos usuarios que no pueden verlas.
- Estructura básica de la página web: la estructura del sitio debe simplificar el acceso al contenido para el usuario que no puede acceder al mismo de manera visual.
- Uso de encabezados: utilizar encabezados conservando una jerarquía lógica, de manera que los lectores de pantalla puedan navegar facilitando el acceso a la web.
- Contraste del texto: el contraste entre el texto y el fondo debe ser de al menos 4.5:1 para texto con tamaño menor a 18 puntos.

Para verificar que el contenido de nuestra página web cumpla con las pautas de accesibilidad, nos hemos asegurado de desarrollar evaluando estos aspectos desde el principio y durante todo el proceso. Además, utilizamos varias herramientas de evaluación de accesibilidad, entre ellas, el servicio de validación de mercado oficial de la W3C (https://validator.w3.org/) y Accesar (http://accesar.onti.argentina.gob.ar/) de la ONTI (Oficina Nacional de Tecnologías de Información).

En base a lo pautado por la cátedra y la propia decisión de considerar la accesibilidad web como un punto esencial en nuestro trabajo, hemos evidenciado que el resultado de la validación de la misma en nuestro proyecto es favorable.

6 Reflexión sobre buenas prácticas en privacidad para el desarrollo de aplicaciones.

Consideramos que en su mayoría hemos tenido en cuenta las buenas prácticas relacionadas a la privacidad en nuestra aplicación. Sin embargo, no nos hemos percatado de ciertos aspectos que creemos que son relevantes a tener en cuenta en el desarrollo de futuras aplicaciones.

El primero de ellos está relacionado con la solicitud de información del usuario que no está destinada a ninguna funcionalidad concreta. Un ejemplo de esto se da con respecto a la solicitud del género del usuario, el cual es un dato que no tiene ningún fin de uso.

Por otra parte, cabe destacar que no se hace distinción respecto de la información que se almacena de un niño o de un adulto. Creemos que deberíamos haber limitado los datos a solicitar, en caso de que se tratase de un usuario menor de edad. Dado que se desconoce la fecha de nacimiento del usuario, podríamos haber implementado una casilla que permita indicar si el mismo es menor de edad.

Por último, al ser un proyecto desarrollado como trabajo integrador de una materia, nuestra app carece de una política de privacidad que indique qué tipo de información se recaba, cómo se usa y con quién se comparte.

7 Conclusión

A lo largo de este informe podemos observar que se tratan aspectos que en la vida cotidiana del desarrollador no reciben la atención que merecen. Estamos acostumbrados a pensar que un programador debe encargarse únicamente de implementar código, pero creemos que es necesario desplazar el enfoque hacia otros aspectos, a medida que avanza el desarrollo de la web.

Hemos mencionado cómo funcionan las PWA, las ventajas que proporcionan los test de unidad, los aspectos a tener en cuenta respecto de la accesibilidad que deberían ofrecer las páginas web, y desarrollamos una breve reflexión acerca de nuestra implementación de las prácticas en privacidad en nuestra aplicación.

8 Bibliografía

- Mounir Lamouri, Rob Dolin. (2013, 17 de diciembre). Web Application Manifest. `https://w3c.github.io/manifest/`
- Smartbear. (s.f). What Is Unit Testing?. `https://smartbear.com/learn/automated-testing/what-is-unit-testing/`
- Ley 26.653 [Senado y Cámara de Diputados de la Nación Argentina]. Accesibilidad de la Información en las Páginas Web. 26 de Noviembre de 2010. `http://servicios.infoleg.gob.ar/infolegInternet/anexos/175000-179999/175694/norma.htm`
- Datos Personales, Política de privacidad, TyC, UNLP, Informática 2022. Mariano Cervellini.