Solución Informática para Optimizar el Trabajo de Producción Agropecuaria de Jóvenes con Discapacidad en Taller Protegido La Plata: una experiencia tecnológica centrada en las personas

Cintia Milagros Valero [1], Matías Omar Batista [1]
{mbatista, cintiavalero} @alu.frlp.utn.edu.ar

Tutores Docentes: Agustín Álvarez Ferrando [1], Leandro Rocca [1], Leopoldo Nahuel [1]
{aaferrando, leorocca, lnahuel} @frlp.utn.edu.ar

[1] GIDAS: Grupo de I&D Aplicado a Sistemas informáticos y computacionales - gidas@frlp.utn.edu.ar
Universidad Tecnológica Nacional - Facultad Regional La Plata

Abstract
El siguiente trabajo estudiantil redacta la experiencia de investigación, desarrollo y transferencia de una aplicación WEB responsive que utiliza servicios de Cloud Computing para la ejecución de procesos y persistencia de datos.
El desarrollo de la solución está siendo co-financiado por fondos de la Provincia de Buenos Aires y de la institución destinataria, una organización no gubernamental destinada a albergar y dar trabajo en el rubro agropecuario a jóvenes con discapacidad.

Palabras Clave
Cloud Computing – Responsive Design - Transferencia Tecnológica – Api REST - IaaS

Introducción

Este trabajo se centra en narrar la experiencia de transferencia al medio socio-productivo, establecida entre el Grupo de Investigación y Desarrollo de la Universidad Tecnológica Nacional - Facultad Regional La Plata (GIDAS), y el Taller Protegido La Plata (TPLP), financiado por el Fondo de Innovación Tecnológica de Buenos Aires (FITBA).

Se conformó un equipo de trabajo integrado por alumnos de 4to año de la carrera Ingeniería en Sistemas de Información, investigadores formados del GIDAS, coordinadores FITBA y personal del TPLP, lo que permitió desarrollar un trabajo estudiantil con el desafío extra de gestionar, no solo recursos tecnológicos, sino también económicos propios de un proyecto software.

Gestión del proyecto

Se realizaron reuniones virtuales de relevamiento a través de la plataforma Zoom [1] para abordar requerimientos con un nivel de frecuencia que fue reduciéndose conforme la comprensión del trabajo fue madurando por parte del equipo GIDAS.

Las sesiones de relevamiento permitieron conocer e indagar sobre los procesos de la organización, terminología propia del dominio, reglas, entre otros.

Se utilizó la herramienta Trello [2] para la asignación y seguimiento de las tareas a desarrollar por cada uno de los miembros del equipo de trabajo.

Relevamiento y modelado de datos

Para llevar a cabo el proceso de relevamiento e identificación de requerimientos se utilizaron las primeras reuniones internas realizadas entre el equipo GIDAS. A medida que se fue profundizando en la comprensión del negocio, el objetivo de las reuniones fue evolucionando hacia el desarrollo de los requerimientos iniciales.

Durante dichas reuniones se exploraron los procesos internos de la organización, sus reglas de operación, y la finalidad de su sistema de administración de gastos. Se realizó un análisis de las planillas de cálculo proporcionadas por el TPLP, donde se identificaron patrones en el almacenamiento

de gastos y en la administración de los recursos entre planillas consecutivas. Se plantearon preguntas y respuestas que permitieron eliminar ambigüedades y pedir información puntual al TPLP para luego obtener requerimientos claros del sistema.

Para la construcción del diagrama de clases se utilizó Enterprise Architect (EA) [3], una herramienta gráfica y dinámica, la cual está enfocada en Lenguaje Unificado del Modelado (UML) que es un lenguaje visual de alto nivel. Este diagrama (Figura 1), fue evolucionando acorde a los nuevos requerimientos que surgían, hasta que finalmente se obtuvo una vista estructurada y coherente que permitió la construcción del modelo de datos y el diseño a bajo nivel de la interfaz gráfica de usuario. La Figura 2 ilustra la evolución del diagrama.

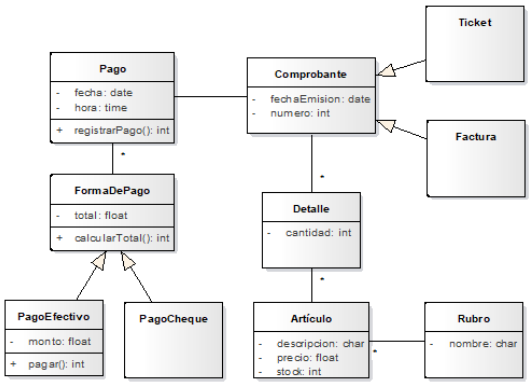

Figura 1: Primera versión del diagrama de clases

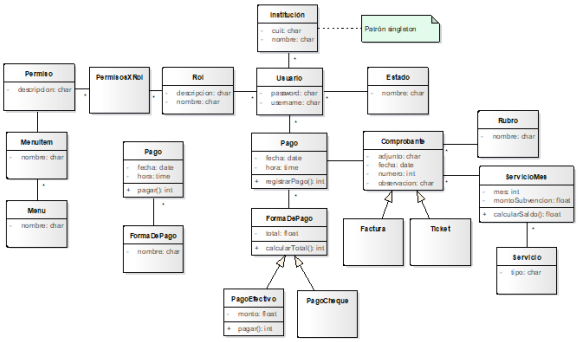

Figura 2: Versión avanzada del diagrama de clases

Para el modelo de datos se utilizó MySQL Workbench v8, una herramienta que permite la representación visual de las entidades y sus relaciones [4].

Tanto el diagrama de clases como el modelo de datos, este último ilustrado en la Figura 3, fueron validados a través de reuniones y discusiones con el equipo, para asegurar su coherencia con los requisitos identificados. Sin embargo, quedó abierto a ajustes y modificaciones que sean necesarios a medida que el desarrollo avance y se

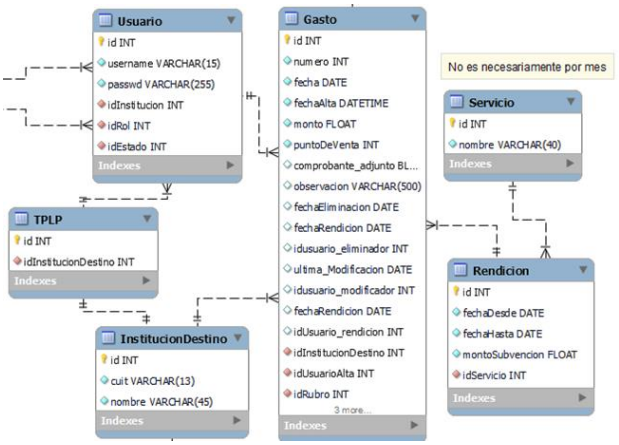

Figura 3: Recorte del modelo de datos

descubran nuevos requerimientos.

Definición de la arquitectura

Para abordar las necesidades del TPLP, se desarrolló una aplicación web utilizando tecnologías tradicionales como HTML, CSS y JavaScript, que consume servicios de una API REST implementada en PHP. Este enfoque está enmarcado dentro de la arquitectura orientada a servicios (SOA) como se visualiza en la Figura 4 y se implementa en la nube a través del modelo de Infraestructura como Servicio (IaaS).

Esta elección arquitectónica se basó en la necesidad de crear una aplicación altamente modular, escalable y adaptable a las cambiantes necesidades del TPLP.

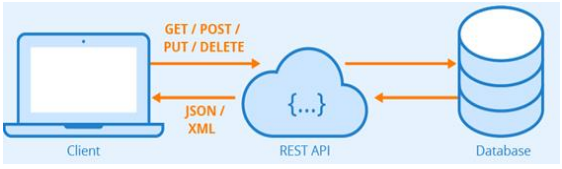

Figura 4: Arquitectura Orientada A Servicios mediante API REST

Arquitectura Orientada a Servicios

La Arquitectura Orientada a Servicios (SOA – Service Oriented Architecture) [5] posibilita la creación del sistema informático destinado a cubrir las necesidades de la institución TPLP a través de la combinación de servicios autónomos e interconectados. Cada uno de estos servicios encapsula una funcionalidad específica, lo que facilita la modularidad y la reutilización de componentes. Al adoptar este enfoque, la aplicación del TPLP se desarrolla como un conjunto interconectado de servicios, como son, la creación, eliminación, modificación y el listado de comprobantes de compra, cada uno enfocado en un aspecto particular de la operación institucional.

API REST

Una API REST se define como un conjunto de reglas y convenciones que facilitan una comunicación estructurada y estandarizada entre la aplicación web desarrollada y el servidor, posibilitando la solicitud y el envío eficiente de datos [6].

Dicha API REST se emplea con el propósito de acceder a recursos específicos en el servidor, llevando a cabo una variedad de operaciones que incluyen la recuperación de información mediante solicitudes GET, la creación de nuevos datos mediante solicitudes POST, la actualización de recursos existentes a través de solicitudes PUT y la eliminación de información mediante solicitudes DELETE. Estas operaciones aseguran una comunicación consistente y eficaz entre la aplicación y el servidor.

La comunicación entre la API REST alojada en el servidor, y la aplicación web como se observa en la Figura 4, se lleva a cabo mediante el formato de intercambio de datos JSON (JavaScript Object Notation), el mismo es un formato de datos ligero y legible que se utiliza comúnmente para transmitir datos entre sistemas en la web, donde los datos se organizan en pares clave-valor y se estructuran en objetos y matrices.

Se optó por la elección de JSON en lugar de XML debido a su simplicidad y eficiencia en la gestión del tamaño de datos. JSON proporciona una estructura de datos más accesible y fácil de comprender, lo que simplifica tanto la comprensión como la depuración del intercambio de información entre la API REST y la aplicación web. Además, JSON tiende a generar datos más compactos en comparación con XML, lo que conduce a una transmisión de datos más rápida y eficiente a través de la red.

La implementación de esta API REST se llevó a cabo mediante el lenguaje de programación PHP.

En la Figura 5 se muestra cómo se compone la estructura del proyecto en Visual Studio Code (VSC) [7].

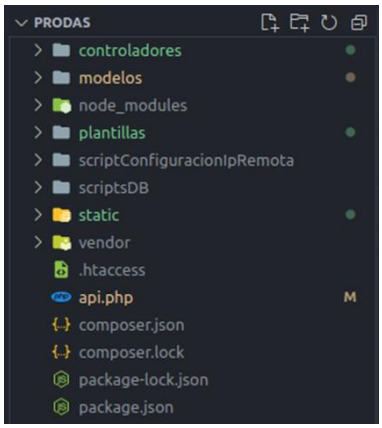

Figura 5: Estructura del proyecto en VSC

El archivo *api.php* se encarga de direccionar las solicitudes HTTP entrantes hacia el controlador correspondiente, en función de la operación solicitada por el usuario.
Dentro de la carpeta *controladores* se encuentra el código que redirige las solicitudes hacia las operaciones implementadas en la carpeta *modelos*. Estos controladores se encargan de aplicar filtros y validaciones con el fin de garantizar el acceso adecuado a las operaciones.

En la carpeta *modelos* se encuentran los Objetos de Acceso a los Datos (DAO) [8], cuya función principal es preparar los datos y atributos necesarios para llevar a cabo la

operación solicitada. Una vez que estos datos están listos, se invocan los métodos pertinentes, permitiendo la ejecución de la operación solicitada.
El archivo *database.php* desempeña un doble rol. En primer lugar, garantiza que las operaciones se realicen de manera segura y evitando posibles ataques. En segundo lugar, proporciona un conjunto de métodos que simplifican las operaciones en la base de datos.

Modelo IaaS
La implementación de la aplicación en la nube a través del modelo IaaS [9] proporciona una serie de ventajas significativas. La infraestructura en la nube ofrece flexibilidad en la asignación de recursos, permitiendo ajustarlos según la demanda en tiempo real. Esta escalabilidad automatizada garantiza un rendimiento óptimo y evita el subaprovechamiento de recursos. Al eliminar la necesidad de mantener servidores locales, se reducen los costos operativos y se elimina la complejidad asociada con la administración de hardware.

Aplicación Web
La decisión de desarrollar una aplicación web se basa en varios factores clave. En primer lugar, una aplicación web ofrece accesibilidad desde cualquier dispositivo con conexión a Internet, lo que resulta de gran utilidad para los usuarios. La aplicación web se desarrolló de manera responsiva, lo que se refiere a la capacidad de la aplicación para adaptarse y funcionar de manera efectiva en diferentes dispositivos y tamaños de pantalla. Esta adaptación redunda en una experiencia de usuario óptima, permitiendo aprovechar al máximo la aplicación en cualquier dispositivo que utilicen. Otra ventaja de la aplicación web es que no requiere instalaciones complejas en dispositivos, lo que simplifica significativamente el proceso de implementación y actualización.
La elección de tecnologías tradicionales para el desarrollo de la aplicación web como los lenguajes HTML, CSS y JavaScript, respaldadas por *frameworks* populares como Bootstrap [10] y jQuery [11], se fundamenta en su amplia adopción y versatilidad en el desarrollo web. HTML permitió definir la estructura de la aplicación, mientras que con CSS se definió su aspecto y presentación visual. Por otro lado, JavaScript permitió añadir dinamismo a la aplicación. La inclusión de Bootstrap y jQuery agiliza aún más el proceso de diseño y desarrollo al ofrecer componentes y funcionalidades predefinidas, contribuyendo así a la eficacia y la calidad general del proyecto.

Persistencia de los datos
Debido a que la aplicación requiere mantener una estructura de datos altamente organizada y relaciones entre diferentes tipos de información, un sistema de gestión de bases de datos relacional se presenta como la elección más apropiada. Se eligió MySQL [12] por su sólida reputación y amplia madurez en la gestión de datos y porque ofrece, además, una solución confiable, segura y robusta para almacenar información.

Autenticación y autorización de usuarios
La autenticación de la aplicación se aborda mediante el uso de JSON Web Token (JWT). Estos tokens proporcionan autenticación y autorización seguras entre los componentes de la aplicación. Al utilizar un sistema de tokens JWT, se logra un acceso seguro a los servicios y se controla la identidad y los privilegios de los usuarios, garantizando la confidencialidad y la integridad de los datos [13].

Diseño y maquetado
La etapa de diseño y maquetado permitió obtener un boceto gráfico y una visión inicial de la usabilidad para la aplicación.

Las reuniones destinadas a discutir la navegación y experiencia de usuario permitieron generar un boceto acorde al diagrama de clases, a las necesidades del negocio, y al tipo de usuario final.

Antes de crear los bocetos se hizo una búsqueda en las redes sociales del TPLP para comprender su identidad visual, y en base a la información pública observada se rearmó un logo existente en su página de Facebook. Para renderizar el logo (Figura 6) se utilizó la herramienta Adobe Illustrator [14], la cual permitió vectorizar y personalizar la imagen encontrada con el objetivo de conseguir una presentación limpia.

Figura 6: Renderizado del logo del TPLP

Después, se definió la paleta de colores (Figura 7) de acuerdo con el rubro al que se enfoca el TPLP. Se utilizó la herramienta Coolors [15], que permite generar, crear y personalizar códigos de colores.

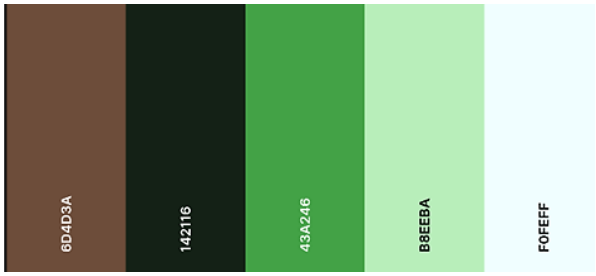

Figura 7: Paleta de colores de la aplicación

Para darle forma a la interfaz de usuario se utilizó la herramienta Balsamiq [16], la cual permitió crear bocetos que muestran la estructura general de la plataforma y la navegación entre los distintos componentes.

Para el maquetado de las interfaces se utilizó el *framework* Bootstrap 5, que permite una maquetación instantánea y responsiva. Los estilos aplicados con CSS se compartieron entre plantillas para reutilizar sus configuraciones. Sin embargo, una importante cantidad de estilos puede ralentizar la descarga de estos. Por ello, se utilizó la minificación, que es una técnica que reduce el tamaño de los archivos aplicando procedimientos como la eliminación de espacios en blanco, comentario, y otros elementos innecesarios sin afectar la funcionalidad del código y lograr una rápida carga [17].

La posibilidad de contar con la navegación entre bocetos permitió abordar con más detalle el desarrollo de la primera versión de la aplicación.

El proceso de diseño se inició con la presentación de la plantilla de inicio de sesión. En principio, se trabajó únicamente sobre una resolución horizontal de escritorio, después se acordó un diseño con enfoque responsivo para permitirle al usuario acceder y consultar gastos con mayor rapidez.

En cuanto a la organización del código CSS, se hizo un análisis sobre algunas metodologías utilizadas en la actualidad y se optó por SMACSS (Scalable and Modular Architecture for CSS) [18]. Esta metodología permite reducir la cantidad de código redundante y asegurar un fácil mantenimiento. Se adaptó esta metodología a las necesidades del proyecto, conservando una plantilla independiente para la paleta de colores, imágenes y tipografías.

Desarrollo de la aplicación

En esta sección se describen las interfaces desarrolladas hasta el momento:

- **Autenticación de usuarios**

Consiste en una interfaz que solicita credenciales de acceso tales como un nombre y una contraseña. Internamente las sesiones se manejan con tokens JWT explicados en secciones anteriores.

- **Alta de gastos**

Desde el menú Gastos → Nuevo gasto, los usuarios encuentran la posibilidad de computar los egresos de dinero del TPLP.

Figura 8: Pantalla de búsqueda de gastos

- **Búsqueda de gastos**

Al ingresar a la aplicación los usuarios se encuentran con una vista predeterminada que muestra los comprobantes del mes actual, tal como ilustra la Figura 8, facilitando así la revisión de los registros más recientes. Esta interfaz también brinda la opción de explorar y buscar comprobantes correspondientes a otros períodos.

- **Alta de rendiciones**

La Figura 9 ilustra la pantalla que permite seleccionar la institución y el servicio a los que corresponde dicha rendición, el subsidio que otorgó la institución y el período en el que se llevó a cabo la rendición.

- **Selección de gastos**

Una vez creada la rendición, la pantalla de selección de gastos permite la selección de comprobantes de gastos que se van a rendir a la institución y servicio correspondiente.

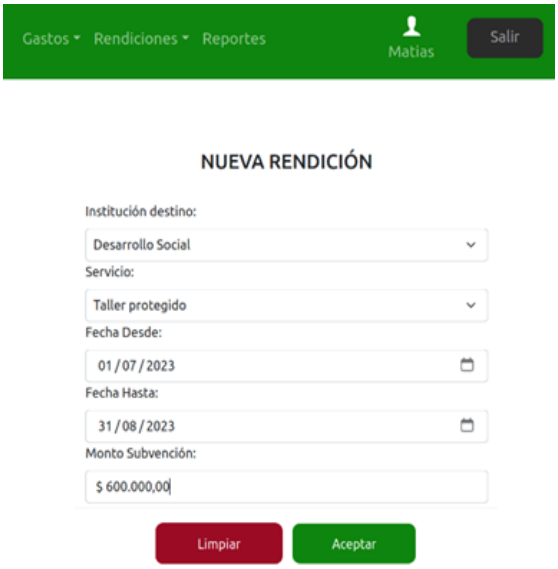

Figura 9: Pantalla para el alta de rendiciones

A continuación, se describen los bocetos diseñados para continuar con el desarrollo de la aplicación

- **Asociación de comprobantes**

La Figura 10 ilustra el boceto pensado para el desarrollo de una interfaz que permita adjuntar, en formato *jpg* o en *pdf*, cada uno de los comprobantes de gastos. Además, la interfaz debe permitir la visualización,

modificación y la eliminación de los comprobantes asociados.

- **Previsualización de comprobantes**

  Esta funcionalidad permite previsualizar los comprobantes adjuntos para que el usuario revise que los mismos sean consistentes con los gastos asociados a la rendición.

Figura 10: Boceto para selección de comprobantes a rendir

- **Resumen de rendición**

Este boceto de interfaz permite la visualización general de la rendición, proporcionando una vista panorámica de la información clave. Una vez que se confirma y verifica la precisión de la rendición, se considera como finalizada, y el sistema redirigirá a una pantalla de descarga.

- **Descarga de rendición**

Esta interfaz permitirá descargar la información de la rendición en formato PDF o Excel, lo que facilita la disponibilidad y el almacenamiento de los datos de la rendición en diferentes formatos. Además, se ofrece la opción de enviar automáticamente un correo electrónico a la institución correspondiente con los datos de la rendición, agilizando así la comunicación y notificación de manera eficiente y práctica.

Trabajos Relacionados

Dado que el presente trabajo estudiantil narra una experiencia de transferencia tecnológica, se describen a continuación algunos ejemplos de transferencias que desde el área de vinculación y transferencia del GIDAS se han desarrollado:

GR – Industria automotriz
La transferencia de conocimientos se basó en la generación de indicadores y la visualización de datos a través de dashboards específicos para toma de decisiones comerciales en una empresa de importante presencia en la industria automotriz de la ciudad de La Plata.

TELEPARK - Tecnologías de Software para Monitoreo de actividades y terapias grupales de bienestar en personas con enfermedad de Parkinson.
Este proyecto se realiza en conjunto con el Programa Permanente de Taller de Pacientes con Parkinson de la Secretaría de Extensión de la Facultad de Ciencias Médicas de la Universidad Nacional de La Plata (UNLP).

CAPNEE - Diseño de Herramienta Computacional que combina componentes de hardware asistivo y software accesible.
Se priorizó la incorporación de pautas universales de accesibilidad para no restringir exclusivamente su desarrollo a un solo tipo de discapacidad.

Conclusión y Trabajos Futuros

Conforme el cronograma FITBA, el equipo GIDAS ha desarrollado los módulos Gestión de Comprobantes y Gestión de Rendiciones. Esta elección estratégica se fundamenta en la relevancia de estos módulos como pilares fundamentales de la funcionalidad del proyecto. El módulo de Comprobantes habilita a los usuarios a realizar operaciones de creación, lectura,

actualización y eliminación de comprobantes de compras. En contraposición, el módulo de Rendiciones simplifica el proceso de presentación de estos comprobantes ante las instituciones pertinentes, asegurando una gestión financiera transparente y efectiva.
La utilización de servicios Cloud Computing fue aceptada por TPLP gracias a las ventajas económicas y de escalabilidad que estas suponen.

El cronograma acordado con FITBA continuará su curso hasta febrero 2024. Quedan por delante desafíos técnicos para el equipo GIDAS. Algunos de estos desafíos consisten en desarrollar los bocetos ya diseñados, como el de adjuntar imágenes de comprobantes de rendición. Para este desafío se viene analizando implementar una aplicación móvil que permita la carga individual de imágenes o documentos adjuntos para cada comprobante, en el mismo acto de registro del gasto.

En este contexto, la próxima tarea consistirá en el desarrollo e investigación de tecnologías y métodos destinados a lograr un almacenamiento y gestión eficientes de los documentos adjuntos relacionados con los comprobantes de compras. Este enfoque enriquecerá considerablemente la funcionalidad de la aplicación.

Referencias

[1] Zoom (Accedido el 20/8/2023)
https://www.zoom.us/
[2] Trello (Accedido el 20/8/2023)
https://trello.com/
[3] Enterprise Architect (accedido el 20/8/2023)
http://www.sparxsystems.com.ar/
[4] MySQL Workbench (accedido el 20/8/2023)
https://www.mysql.com/products/workbench/
[5] SOA (Arquitectura orientada a servicios)
https://www.redhat.com/es/topics/cloud-native-apps/what-is-service-oriented-architecture
[6] API REST (accedido el 22/8/2023)
https://aws.amazon.com/es/what-is/restful-api/
[7] Visual Studio Code (accedido el 20/8/2023)
https://code.visualstudio.com/
[8] Patrón DAO (accedido el 23/8/2023)
https://www.geeksforgeeks.org/data-access-object-pattern/
[9] Modelo IaaS (accedido el 23/8/2023)
https://cloud.google.com/learn/what-is-iaas?hl
[10] Bootstrap (accedido el 23/8/2023)
https://rockcontent.com/es/blog/bootstrap/
[11] JQuery (accedido el 24/8/2023)
https://es.wikipedia.org/wiki/JQuery
[12] MySQL (accedido el 24/8/2023)
https://es.wikipedia.org/wiki/MySQL
[13] JWT (accedido el 25/8/2023)
https://jwt.io/introduction
[14] Adobe Illustrator (accedido el 26/8/2023)
https://www.adobe.com/la/products/illustrator.html
[15] Coolors (accedido el 26/8/2023)
https://franciscotorreblanca.es/coolors-herramienta-digital-generar-paletas-de-color/
[16] Balsamiq (accedido el 27/8/2023)
https://www.isdi.education/es/blog/balsamiq-herramienta-para-realizar-prototipos-de-tus-proyectos
[17] Minificación (accedido el 28/8/2023)
https://www.suratica.es/que-es-la-minificacion/
[18] SMACSS (accedido el 29/8/2023)
https://www.toptal.com/css/smacss-scalable-modular-architecture-css