Sumario

Sumario

Documento en revisión, podra cambiar la puntuación del primer problema y la puntuación y contenido de los siguientes

Modificaciones realizadas en este documento

Creación 6/1/2024

- Creación del documento. (Incompleto)

Modificación 9/2/24

- Se ha modificado la ponderación de los RA para que sean: RA-9 (55%), RA-7 (30%) - RA-8(15%)
- Definición completa del problema 3
- Indicar en problema 3.1 que es obligatorio el uso de HttpClient

- Detallada toda la puntuación y ponderación de resultados de aprendizaje

Normas de obligado cumplimiento

Para la realización de la aplicación **será obligatorio el uso del framework Laravel**.

El acceso a la base de datos se realizará utilizando los mecanismos de acceso a datos de los que disponga el framework. Se deberá utilizar obligatoriamente Eloquent para manejar la base de datos incluyendo la **lógica de consulta en el módelo o clases auxiliares (véase Laravel: Más allá de CRUDs)**, **NO en los controladores**.

Los problemas deberán ser documentados con comentarios, indicando en cada fichero con código php el autor, la fecha de creación, la versión del fichero.

Se documentará cada una de las funciones y variables de clase que se creen utilizando ela propuesta de estándar PSR-5: PHPDoc (Wikipedia) del cual podréis obtener más información en este artículo.

Básicamente lo que debéis hacer es abrir comentarios de tipo bloque `/** .... */` delante de cada función y variable de clase, y el entorno (eclipse, netbeans) se encargará de generaros automáticamente las etiquetas (`@...`) pertinentes. Si trabajáis con Visual Studio code esta extensión PHP DocBlocker os resulte interesante.

Se generará la documentación de nuestro programa usando alguno de los programas existentes de generación automática como: ApiGen, Doxygen, o phpDocumentor

Se verificará el funcionamiento de la aplicación en un contenedor Docker o servidor Linux. Para ello se os proporcionará llegado el momento un espacio web y una base de datos con la que podréis trabajar trantando de simular un entorno lo más real posible.

Recordad que en un entorno Linux los ficheros distinguiran entre mayúsculas y minúsculas.

Se recomienda que vayáis probando vuestro proyecto según lo vayáis creando y no dejéis esta operación para el último día.

Comentarios

Cada problema de los expuestos a continuación se creará en una carpeta independiente, que llamaréis “Problema 1”, “Problema 2”, etc.

Cada carpeta que contiene los problemas tendrá la siguiente estructura:

Tipo	Elemento	Descripción
	¿?	Carpetas que utiliza el framework
	 Assets	Imagenes, ficheros de estilo (css), fichero de script (js), y otro tipo de

	css img js	ficheros.
	...	
	install	En esta carpeta se incluirá cualquier información que se consideré relevante para instalar la aplicación. Sería interesante que incluyérais un instalador. Aplicación que cree la base de datos y configure el resto de parámetros necesarios.
	bd.sql	Fichero que contiene un script sql que crea la base de datos que utilizará el programa. En el script se creará - La base de datos - La estructura de las tablas - Los usuarios si utilizase algúno diferente
	Doc	Documentación de la aplicación.

Los problemas tienen como propósito obligaros a que trabajéis los contenidos vistos en clase así como que profundicéis en vuestros conocimientos sobre la metodología de la programación (como resolver los problemas). Esta práctica está pensada para que la realicéis en clase completando el trabajo en vuestra casa. Ante cualquier duda acerca de cómo afrontar un problema deberíais preguntar al profesor al respecto para que el os oriente y no perder tiempo con propuestas estructuralmente erroneas..

Cada problema tendrá una fecha de entrega, fecha en la cual debereríais tener completado el ejercicio, esta fecha es orientativa y tiene por objeto evitar que os durmáis en los laureles, tan solo es obligatorio tenerlo todo completo en la fecha de entrega.

Los problemas, **obligatoriamente deberéis enseñarselos al profesor funcionando en clase antes de la fecha tope de entrega**.

Calendario

Enero o febrero	Recuperación de alumnos que tienen pendiente la primera evaluación – PHP. Entrega de la práctica de la 1ª Ev. – Concretar con alumnos--
lunes 4 de marzo	Entrega -completa- de todas las prácticas de la 2ª Ev. Último día disponible si hay contratiempos el lunes 7 de marzo.
Miércoles 13 de	Sesión de evaluación de la 2ª Evaluación

marzo	
4-13 de marzo	Corrección de la práctica y examenes (tareas) de recuperación que se precisen.

Para el desarrollo de está aplicación será ***obligatorio*** *que utilicéis un repositorio de código GIT público (GitHub u otros) que cada usuario creará utilizando las herramientas que disponga vuestro entorno de desarrollo para trabajar con el mismo. Se tratrá de utilizar las herramientas que proporciona nuestro entorno de desarrollo para sincronizar dicho repositorio.*

La aplicación debe ser mostrada al profesor funcionando en un contenedor docker, en el espacio web del instituto o en un espacio web que os indique vuestro profesor. Se recomienda que se vaya probando en unas condiciones similares al servidor de producción a medida que se va realizando para evitar problemas de última hora.

1 – Gestión de Incidencias/tareas en Laravel

Todo el código implementado debe realizarse utilizando el framework laravel utilizando los mecanismos que proporciona para realizar las distintas operaciones.

Especificaciones

Las especificaciones siguientes puede que estén incompletas, tan solo son una recopilación de las necesidades iniciales que nos ha transmitido el cliente.

La empresa de mantenimiento de ascensores Nosecaen S.L. recientemente ha sido adquirida por la multinacional SiempreColgados. Dicha empresa visto el nivel de productividad que tenían en la empresa absorbida han decido implantar el modelo de trabajo de la empresa Nosecaen y utilizar los modernos sitemas de digitalización empleados en la misma. Aunque debido a su volumen ententiende que la aplicación de incidencias debe ser reformada para permitir nuevos requisitos. El departamento de informática después de un estudio detallado con el nuevo equipo directivo nos ha trasladado los requisitios que debe cumplir la aplicación

Se desea realizar una aplicación web que nos permita gestionar las incidencias y ordenes de trabajo de los operarios de nuestra empresa atendiendo a los siguientes requisitios:

- La aplicación llevará un registro de empleados, que serán los que puedan acceder a la aplicación. De dichos empleados guardaremos la siguiente información:
  - DNI
  - Nombre
  - Correo

- Teléfono
- Dirección
- Fecha alta
- Tipo: Operario o Administrador
- *-otros datos que se precisasen-*

Los empleados de tipo administrador podrán gestionar la lista de empleados. Cada empleado a su vez podrá modificar sus datos de contacto Correo y fecha de alta.

- La empresa tendrá una cartera de clientes registrados, sobre los cuales cobraremos unas cuotas mensuales. Dichos clientes serán gestionados por los empleados de tipo administrador. Para cada cliente guardaremos la siguiente información
  - CIF
  - Nombre
  - Teléfono
  - Correo
  - Cuenta corriente
  - Pais: Seleccionar de la lista de paises que tenéis en la tablas proporcionadas en temas anteriores
  - Moneda
  - Importe cuota mensual
  - *-otros datos que se precisasen-*
- Mensualmente se hará un cargo a nuestros clientes por el servicio de mantenimiento que prestamos. Dicho cargo se almacenará como una cuota en la que guardaremos la siguiente información. Esporadicamente también se podrán anotar cuotas por trabajos excepcionales realizados
  - Concepto
  - Fecha emisión
  - Importe
  - pagada (S/N) – *Puede que no se precise -*
  - fecha de pago
  - notas
  - *-otros datos que se precisasen-*
- Los administradores podrán realizar las siguientes operaciones en el programa

- Gestionar empleados: añadir, dar de baja, cambiar el tipo y los otros datos, etc.
- Gestionar clientes: añadir, dar de baja
- Gestionar las cuotas: añadir remesa mensual a todos los clientes, añadir cuota excepcional, corregir cuota, borrar cuota, ver lista de cuotas, etc.
- Crear factura de la cuota cargada para enviar al cliente.
- Enviar de forma automáticamente por correo factura en formato PDF de factura al cliente.

l Además nuestra aplicación al igual que la anteriormente realizada permitirá todo lo que realizaba anteriormente, con la restricción de que ahora

- Un operario solo verá las tareas que tiene asignadas
- El administrador asignará un empleado de la lista de operarios que trabajan en la empresa a través de un campo de selección. **No se permitirá** crear tareas sin asignar operario a los administradores

l Los clientes podrán registrar una incidencia introduciendo los mismos datos que un administrador con la restricción de que no asignarán el operario que realizará dicha tarea. En este caso quedará sin marcar y luego será un administrador el que tendrá que realizar dicha asignación.
Para garántizar la identidad de los clientes y que terceros no realicen anotaciones que no proceden se solicitará al cliente que introduzca su CIF y teléfono, los cuales deben coincidir con los registrados

Requisitos anteriores

Nuestra aplicación web nos permitirá realizar las siguientes operaciones:

- Ver la lista de incidencias/tareas. (*Adm. y ope.*)
- Añadir una nueva indicencia/tarea. (*Adm.*)
- Modificar datos de una indicencia/tarea. (*Adm.*)
- Eliminar una tarea. Confirmando la operación para evitar errores. (*Adm.*)
- Cambiar el estado de una incidencia/tarea
  Completar una tarea incluyendo anotaciones si se precisan (*ope.*)
- ~~Buscar o filtrar tareas utilizando distintos campos. (*Adm. y ope.*)~~ No obligatorio

La información que almacenaremos sobre las tareas será la siguiente:

- ***Cliente que encarga el trabajo*** *(nuevo)*
- *Persona de contacto: Nombre y apellidos de la persona.*
- *Teléfono/s contacto: Nº de telefono de contacto de la persona de contacto.*
- *Descripción:* Texto descriptivo identificativo de la tarea
- *Correo electrónico:* Correo electrónico de la persona de contacto.
- *Dirección*: Dirección donde debemos hay que ir a realizar la tarea.
- *Población*
- *Código postal*

- *Provincia*: Para este campo utilizaremos un <select>. En este campo se almacenará un código numérico acorde a las indicaciones del INE. Dicho código será el mismo que que los dos primeros dígitos del código postal.
- *Estado*: Estado en el que se encuentra la tarea (P=Pendiente, R=Realizada, C=Cancelada, ...)
- *Fecha de creación de la tarea*: Fecha en la que se ha creado la envío. Este campo se generará automáticamente. *Se debería usar un disparador de la base de datos*.
- *Operario encargado:* Nombre o identificación del operario encargado de la realización de la tarea
- *Fecha de realización*: Fecha en la que se realizará la tarea.
- *Anotaciones anteriores*: Cualquier texto que se desee incluir para explicar el trabajo a realizar.
- *Anotaciones posteriores*: Anotaciones realizadas por los operarios después de realizar la tarea.
- Fichero resumen de tareas realizadas. Para cada tarea realizada se permitirá que el operario pueda adjuntar un fichero con indicaciones del trabajo realizado. Dicho fichero ser almacenará en una carpeta dentro del servidor y no será accesible fácilmente.

La información que contiene estos campos debe cumplir:

- Los campos descripción y persona de contacto debe tener algún valor
- El teléfono de contacto debe tener un valor y si existe debe tener un formato válido, sólo números, y caracteres de separación (espacio, guión, y otros que estiméis oportuno).
- El código postal, si existe, debe tener un formato válido, 5 números.
- La provincia debe ser algúna de las existentes en españa. Se debe permitir seleccionar la provincia de una lista deplegable.
- El correo electrónico es obligatorio y debe tener un formato correcto.
- La fecha de realización debe tener un formato válido y ser posterior a la fecha actual. Se debe admitir una cadena con el formato d/m/aa.
- La fecha de creación no se podrá modificar, aunque si aparecerá en las consultas y modificaciones.

Nota:

- Se recomienda que utilicéis como iconos los proporcionados por la página FontAwesome
- *Se se recomienda la utilización de framewos de CSS como Bootstrap o Materialize.*
- *Más adelante, en el problema 3, intentaremos utilizar Quasar Framework el cual está desarrollado siguiendo las directrices de Material Design, por lo que se recomienda utilicéis Frameworks de CSS que sigan estas directrices.*

Antes de comenzar con la programación se recomienda que reviséis vuestro modelo de datos con el profesor para ver si tiene alguna carencia estructural difícil de subsanar más adelante.

Problema 2 – Pruebas

Para el programa realizado en el problema 1 deberemos elaborar una **bateria de pruebas** que

verifique el correcto funcionamiento del mismo. Estas pruebas deben ser **automátizadas** y deberán probar al menos

- Que todas las rutas existan y muesten algo.
- Para alguno formularios deberemos probar su envío y procesado

Utilizremos los mecanismos que proporciona Laravel, aunque se pueden investigar e incluir otros que serán igualmente valorados.

Vease también:

- Pest | The elegant PHP testing framework (pestphp.com)
- Primero pasos con PEST el Framework de pruebas de PHP (daguilar.dev)

Problema 3– Servicios

Debemos ampliar nuestro programa para permitir para que incluya las siguientes funcionalidades

3.1 Cliente servicio - Importe en moneda de cada Pais

Debido a la expansión internacional nos indican que las facturas deben ser emitidas en la moneda local de cada pais. Esto genera problemas de facturación debido a la fluctuación de las monedas y nuestro departamento de contabilidad nos solicita que debemos indicarles en euros el importe que hemos cobrado de nuestros clientes.

Para esto debemos tener en cuenta lo siguiente:

- Los importes registrados con cada cliente están indicados en la moneda local
- El importe efectivo que consideraremos para nuestros cálculos será el que corresponda al convertido en euros el día que se ha pagado la factura.
- Se incluirá una opción de marcar factura como pagada en cuyo caso se incluirá el importe en euros que se ha registrado contablemente.

Nota: tengase en cuenta que puede precisarse añadir campos a nuestro modelo de datos para solventar este problema.

Para saber el cambio que hay cada día podremos utilizar cualquer API Pública que conozcamos o servicio que encontremos. Se deja a continuación alguna sugerencia:

- Currency-api: Free Currency Exchange Rates API with 150+ Currencies & No Rate Limits
- National Bank of Poland: collection of currency exchange rates (data in XML and JSON)
- Czech National Bank
- Currency Exchange: Lista de API

Para consultar la API pública o el fichero con datos OBLIGATORIAMENTE deberemos usar HttpClient

3.2 Servicio REST – Creación y documentación

Para alguna de las tablas de vuestra aplicación crearéis un servicio Rest el cual consumiréis usando javascript o alguno de los programas / extensiones disponibles para hacer pruebas y consumir servicios REST como:

- Postman
- VS Code – REST Client (Instrucciones - How to test REST API with Visual Studio Code — REST Client extensions)

Vuestro servicio deberá utilizar como lenguaje de datos el formato Json, o XML.

Obligatoriamente deberéis utilizar el mecánismo Route::apiResources de enrutamiento que nos proporciona Laravel.

Se valorará positivamente la utilización de códigos de respuesta en las peticiones acorde a la especificación de REST como podéis consultar en - HTTP Status Codes

Youtube - Consumir una API Rest en Laravel mediante una aplicación en

Youtube - Valida los Request de tu API REST en Laravel

Youtube - API REST CRUD con LARAVEL

Laravel API: Crea y Prueba una API en Laravel

Deberéis documentar el servicio utilizando la especificación OpenApi (Swagger), para que sea de fácil uso por los clientes. Para ello podréis utilizar si lo consideráis oportuno alguno de los paquetes que existen en Laravel que facilitan el trabajo.

Cómo documentar una API en Laravel usando Swagger
https://styde.net/como-documentar-una-api-en-laravel-usando-swagger/
OpenAPI Initializer
https://laravel-news.com/openapi-initializer
Laravel OpenAPI
https://vyuldashev.github.io/laravel-openapi/

Documentos interesantes:

- ¿Qué es OpenAPI?
  https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-openapi/
- Introducción a OpenAPI: ventajas e implementación
  https://www.alten.es/introduccion-a-openapi-ventajas-e-implementacion/
- Laravel API Documentation with OpenAPI/Swagger
  https://blog.quickadminpanel.com/laravel-api-documentation-with-openapiswagger/
- OpenAPI 3.0 Tutorial
  https://support.smartbear.com/swaggerhub/docs/en/get-started/openapi-3-0-tutorial.html
- Swagger Editor

https://editor.swagger.io/

- OpenAPI Specification
  https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.0.2.md
  https://swagger.io/specification/

3.3. Validar el usuario utilizando los servicios de Facebook, Google, Twitter u otros

Utilizando OAuth se desea que vuestra aplicación permita registrarse a los usuarios obteniendo las credenciales de Facebook, Google u otros. Para esto utilizaremos el paquete Laravel Socialite

Véase:

Styde.net - Autenticación mediante Facebook con el paquete Laravel Socialite

3.3. Pago de nuestros productos con tarjeta – Niv. Dif: Avanzado

Si alguien desea investigar, que implemente la funcionalidad de pago con PayPal o con tarjeta a través de un TPV virtual (esto será más complejo, pues hay que solitarlo al banco).

Este apartado no tiene por que estar completamente operativo, podéis informaros al respecto e intentar implementar algo, aunque es complicado su integración completa con vuestra aplicación. Además precisáis realizar las pruebas desde un servidor público en algunos casos.

Lo que se pretende es que al cliente se le de una opción desde la que poder pagar su cuota, y que este quede marcada como pagada a través de un TPV o una pasarela de pago. No es preciso que trabajéis la interfaz, servirá con que permitáis seleccionar una cuota no pagada desde la ventana de cuotas e iniciar el proceso. En una situación real será diferente pero para nuestros propósitos nos sirve igual.

Información sobre TPV Virtual

- TPV Virtual: qué es, qué me cuesta y cuál es el mejor
- Como implementar una pasarela de pago mediante tarjeta de crédito utilizando PHP
- Cómo integrar una pasarela de pago TPV RedSys con PHP
- Nueva Web con TPV Redsys y control de pagos

Uso de PayPal

PayPal dispone de un SandBox desde el que se pueden simular compras sin realizar pagos reales.

- HOW TO SET UP PAYPAL INTEGRATION WITH PHP & MYSQL
- PayPal Sandbox Testing Guide
- PayPal Checkout Demo / PayPal Checkout Overview / Code Samples Java – PHP
- Laravel: Cómo integrar pagos con PayPal

Problema 4 – Generación dinámica de páginas Web interactivas

Para evitar problemas puede que os interese probar estas funcionalidades aparte de vuestro proyecto original. No es necesario que esté en el mismo proyecto, podéis integrarlo en otro u otros sin problema.

4.1 Integración de librerias de javascript utilizando CDN - 1 pto.

Se desea ampliar el programa (repetir uno de los CRUD) para que gestione la interacción con el usuario (creación/modificación/borrado) utilizando javascript y sin recargar de nuevo toda la página para cada uno de las operaciones.

Se deberán validar los campos utilizando javascript (antes de enviar) y si es posible integrando la validación con la realizada en el servidor (resultado de una operación no válida).

Para este problema podríais hacer uso de DataTable

4.2Uso de Vue/Quasar utilizando CDN - 1 pto.

Se desea ampliar el programa (repetir uno de los CRUD) para que gestione la interacción con el usuario (creación/modificación/borrado) utilizando vue/quasar y sin recargar de nuevo toda la página para cada uno de las operaciones. Se recomienda el uso de la librería Axios para realizar las peticiones.

Se deberán validar los campos utilizando javascript (antes de enviar) y si es posible integrando la validación con la realizada en el servidor (resultado de una operación no válida).

Véase:

- Quasar UMD - CDN install

4.3 Uso de Vue utilizando VITE y componentes .vue - 1 pto.

Se desea ampliar el programa (repetir uno de los CRUD) para que gestione la interacción con el usuario (creación/modificación/borrado) utilizando vue y sin recargar de nuevo toda la página para cada uno de las operaciones.

Se deberá elaborar la parte cliente utilizando componentes de Vue que serán compilados utilizando Vite.

Se deberán validar los campos utilizando javascript (antes de enviar) y si es posible integrando la validación con la realizada en el servidor (resultado de una operación no válida).

Véase:

- Setting up Laravel with Inertia.js + Vue.js + Tailwind CSS
- Laravel Vuejs Crud SPA using Inertia Js

- InertiaJS-Demo application
- The Ultimate Guide to Inertia.js / La Guía Definitiva de Inertia.js
- Build Modern Laravel Apps Using Inertia.js

Comentarios y evaluación

- La nota de la práctica supondrá el 50% o más de la calificación sobre el bloque de contenido evaluado, por lo que su entrega será obligatoria.
- La copia de todo o parte del ejercicio supondrá la inmediata eliminación de la parte copiada. Se dividirá la nota de los implicados entre el número de copias. Igualmente en caso de identificar a la persona que ha copiado se le penalizará en su calificación.
- Cada alumno enseñara individualmente el funcionamiento de la aplicación al profesor y explicará la organización del código creado, pudiendo solitiarse realizar cambios en la misma.

Puntuación del proyecto

Con nuestro proyecto pretendemos garantizar que cumplis los requisitos mínimos exigidos en el curriculo, que se establecen a través de Resultados de aprendizaje.

Nota: Los porcentajes en los apartados indican la ponderación sobre la nota que se obtendrá en cada RA. No podrá superar el 100% aunque la suma de los mismos lo permita

	Resultados de aprendizaje y C.E.	**Apartados asociados**
UT 9 55% RA 9 4 ptos	Desarrolla aplicaciones Web híbridas seleccionando y utilizando librerías de código y repositorios heterogéneos de información. • *Se han reconocido las ventajas que proporciona la reutilización de código y el aprovechamiento de información ya existente.* • *Se han identificado librerías de código y tecnologías aplicables en la creación de aplicaciones Web híbridas.* • *Se ha creado una aplicación Web que recupere y procese repositorios de información ya existentes.* • *Se han creado repositorios específicos a partir de información existente en Internet y en almacenes de información.* • *Se han utilizado librerías de código para incorporar funcionalidades específicas a una aplicación Web.*	Requisito - Enseñar proyecto usando docker o servidor linux 15% 1.1 Gestionar incidencias/tareas por administrativos: CRUD 15% 1.2 Gestionar empleados por administrativos CRUD 10% 1.3 Gestionar clientes por administrativos CRUD (0,5) 10% 1.4 Gestionar incidencias/tareas por operarios 10% 1.5 Gestionar cuotas - CRUD 10% 1.5 operaciones adionales de filtrado (5%) 10% 1.6 - Crear cuota mensual para todos los clientes 10% 1.9 Crear factura en pdf para cuota – Mostrar/descargar (5%+5%)

	Resultados de aprendizaje y C.E.	Apartados asociados
	• *Se han programado servicios y aplicaciones Web utilizando como base información y código generados por terceros.* • *Se han probado, depurado y documentado las aplicaciones generadas.*	10% 1.10 Registrar incidencia por parte de los clientes y operaciones asociadas para los administrativos(0,5) 20% 2. Pruebas 0.1 Uso de Github 0.2 Documentación
UT 7 30% RA 7 3 ptos	RA7. Desarrolla servicios Web analizando su funcionamiento e implantando la estructura de sus componentes. • *Se han reconocido las características propias y el ámbito de aplicación de los servicios Web.* • *Se han reconocido las ventajas de utilizar servicios Web para proporcionar acceso a funcionalidades incorporadas a la lógica de negocio de una aplicación.* • *Se han identificado las tecnologías y los protocolos implicados en la publicación y utilización de servicios Web.* • *Se ha programado un servicio Web.* • *Se ha creado el documento de descripción del servicio Web.* • *Se ha verificado el funcionamiento del servicio Web.* • *Se ha consumido el servicio Web.*	Requisito - Enseñar proyecto usando docker o servidor linux 10% 1.8 Enviar correo informativo (con o sin pdf) a clientes cuando se crea cuota 20% 3.1 Cliente servicio - Importe en moneda de cada Pais 10% 3.2 Servicio REST – Creación 10% 3.2 Servicio REST – Gestión errores 10% 3.2 Documentación servicio con Swagger 10% 3.2 Pruebas del servicio con postman 10% 3.2 Pruebas del servicio con PHPUnit Laravel 10% 3.2 Cliente (SPA javascript) que consume el servicio 15% 3.3. Validar el usuario utilizando los servicios de Facebook, Google, Twitter u otros 25% 3.3. Pago de nuestros productos con tarjeta – Niv. Dif: Avanzado (No es preciso completarlo, puede estar planteado y se valorará). 0.1 Uso de Github 0.2 Documentación
UT 8 15% RA 8 3 ptos	RA8. Genera páginas Web dinámicas analizando y utilizando tecnologías del servidor Web que añadan código al lenguaje de marcas. • *Se han identificado las diferencias entre la ejecución de código en el servidor y en el cliente Web.* • *Se han reconocido las ventajas de unir ambas tecnologías en el proceso de desarrollo de programas.*	Requisito - Enseñar proyecto usando docker o servidor linux 20% Uso general de javascript en proyecto. Depuración en cliente, etc. 30% 4.1 Integración de librerias de javascript utilizando CDN 30% 4.2 Uso de Vue/Quasar utilizando CDN 30% 4.3 Uso de Vue utilizando VITE y componentes .vue

	Resultados de aprendizaje y C.E.	Apartados asociados
	• *Se han identificado las librerías y las tecnologías relacionadas con la generación por parte del servidor de páginas Web con guiones embebidos.* • *Se han utilizado estas tecnologías para generar páginas Web que incluyan interacción con el usuario en forma de advertencias y peticiones de confirmación.* • *Se han utilizado estas tecnologías, para generar páginas Web que incluyan verificación de formularios.* • *Se han utilizado estas tecnologías para generar páginas Web que incluyan modificación dinámica de su contenido y su estructura.* • *Se han aplicado estas tecnologías en la programación de aplicaciones Web.*	0.1 Uso de Github 0.2 Documentación

Criterios de evaluación

La corrección y puntuación de cada apartado se realizará atendiendo a los siguientes parámetros:

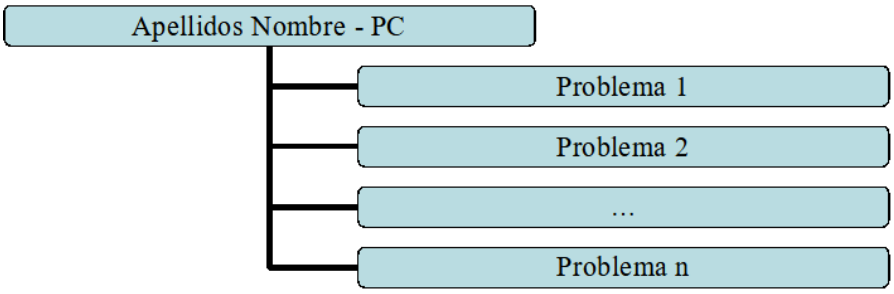
- Funcionalidad: Que el programa realice lo que se pide
- Interfaz: Que el programa que utilicen recursos gráficos que faciliten la interacción con el usuario y se usen los recursos que proporciona el entorno de desarrollo
- Estilo de programación: que el código del programa sea facilmente entendible y modificable por otras personas. Para ello deberá regirse por las directrices de la programación estructurada, la programación orientada a objetos y utilizar patrones de diseño de software.

Instrucciones para entregar la práctica

La práctica se subirá comprimida al aula virtual en una tarea habilitada al efecto.

Dentro del fichero comprimido incluiréis el código fuente y ficheros de proyecto de vuestro programa. Los cuales están contenidos en la carpeta del proyecto.

El fichero estará organizado por carpetas siguiendo la siguiente estructura



Esta práctica deberá ser publicada en el dominio del centro, en el espacio que se os ha reservado al efecto.