Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas

Ingeniería de Software

4CM50

Elaborado por:

- Pérez Gutiérrez Miguel Angel – Líder
- Mendoza Espinoza Karla Daniela – BD
- Olmedo Chávez Evelyn Odete - Diseñador
- Altamirano Garcia Yazmín – Tester
- Galán García Luis Jesús – Arquitecto
- Rojas Ramos Manuel Antonio - Desarrollador
- Amador Ambriz Roberto Iran – Análisis

22 de junio del 2021.

INDICE

REQUERIMIENTOS PARA EL “SISTEMA DE VIATICOS” ORIGINALES

Una empresa requiere un Sistema de gestión de viáticos, con el objetivo de satisfacer las siguientes necesidades:

- Que los empleados puedan registrar sus comprobantes desde el lugar de la comisión.

- Que se realicen solicitudes de viáticos y de transporte.

- Que las solicitudes pasen por un flujo de aprobación.
  Jefe -> área financiera -> Autorización

- Se debe llevar un control presupuestal por área.

- Los empleados deberán comprobar el 80% de sus gastos mediante facturas.

- El presupuesto deberá estar calendarizado y se dividirá por partida y por
  Partida para viáticos y transporte

- El Sistema deberá permitir elaborar un reporte de actividades referente a la comisión.
  Enfocarse en el registro
- Todas las solicitudes, aprobaciones y validaciones del Sistema se deberán realizar con una aprobación electrónica.
- Enfocarse en el registro
- Los reportes que genera el sistema son:
  - Historial de viáticos por empleado
  - Gastos desglosados por empleado *
  - Comisiones asignadas por empleado *
  - Presupuesto ejercido por área
  - Presupuesto asignado por área

MODELO DE ESTIMACIÓN

Que los empleados puedan registrar sus comprobantes desde el lugar de la comisión.	10 hrs	5 días
Que se realicen solicitudes de viáticos y de transporte.	10 hrs	5 días
Que las solicitudes pasen por un flujo de aprobación	20 hrs	10 días

Se debe llevar un control presupuestal por área.	10 hrs	5 días
Los empleados deberán comprobar el 80% de sus gastos mediante facturas	15 hrs	7-8 días
El presupuesto deberá estar calendarizado y se dividirá por partida y por área.	20 hrs	10 días
El Sistema deberá permitir elaborar un reporte de actividades referente a la comisión.	10hrs	5 dias
Todas las solicitudes, aprobaciones y validaciones del Sistema se deberán realizar con una aprobación electrónica.	15 hrs	7-8 dias
Los reportes	20+ hrs	10 dias

Reglas de Negocio

- ✓ Los viáticos podrán ser nacionales e internacionales y se asignarán dependiendo al puesto del empleado. Restricciones a partir del puesto
- ✓ Los viáticos se depositarán 24 horas del inicio de la comisión en moneda del país de origen. avisar al empleado
- ✓ Las monedas que manejara el sistema son: dólares, euros y pesos
- ✓ Los empleados tendrán hasta 5 días hábiles a la conclusión de su comisión para realizar la comprobación de viáticos. Notificación de plazo de vencimiento al administrador y empleado
- ✓ En caso de que el área no cuenta con disponibilidad presupuestal, se tendrá que hacer una adecuación presupuestal.
- ✓ El presupuesto se tomará por área y por mes, no se podrá tomar presupuesto de otras áreas, ni de meses posteriores. Meses anteriores y posteriores
- ✓ Las facturas que se carguen al Sistema deberán validar que la extensión sea XML.
- ✓ Las solicitudes de los viáticos se tendrán que gestionar con 2 días natural.

ADMINISTRADOR DE LA TECNOLOGÍA A UTILIZAR:

Opción 1:

Front-End – Angular

Se realizo la elección de esta tecnologia tomando en cuenta principalmente el tiem po que tendriamos disponible para el desarrollo de proyecto. Angular nos da la ventaja de que promueve una arquitectura ya establecida, por lo que podemos c

entrarnos en solo implementar las funcionalidades y el desarrollo de interfaces, lo que agilizara nuestros tiempos de trabajo.

Back-End - Node JS

Se realizo la elección de esta tecnologia tomando en cuenta principalmente el tiem por que tendríamos disponible para el desarrollo de proyecto. Al tener un tiempo relativamente corto para el desarrollo de este proyecto, Node.js, nos da la posibilidad de agilizar los tiempos de trabajo, dado a que es un framework de JavaScript, lenguaje que es interpretado y al tener la posibilidad de usar distintos modulos para ciertas funcionalidades sin la necesidad de implementarlos nosotros mismos podemos reducir considerablem ente el tiempo de trabajo enfocandonos en la funcionalidad general de nuestra aplicación.

MySQL

Elegimos este sistema de base de datos ya que vamos a trabajar con datos estructurados y Mysql ofrece almacenamiento en tablas relacionadas entre sí. Así mismo, cuenta con un gestor Workbench que nos ayuda a tener un mejor control de datos y de la estructura de las bases dando mayor velocidad y flexibilidad al trabajo. Además que la conexión es sencilla y la interacción también.

ARQUITECTURA A IMPLEMENTAR EN EL PROYECTO

Como equipo hemos decidido implementar la arquitectura de tres capas debido a que la arquitectura de microservicios es muy ambiciosa para el alcance de nuestro proyecto, decidimos usar la arquitectura de tres capas para mantener separados los componentes, presentación, lógica de negocios y el acceso a datos de nuestro sistema, garantizando así una arquitectura escalable y fácil de manejar y que podemos ir tomándola como guía sin mucha complicación en el proceso de desarrollo del proyecto.

REFINANCIAMIENTO DE REQUERIMIENTOS

Historia

HU1. Como usuario debo poder ingresar al sistema en base a mi rol.

HU2. Como empleado debo poder visualizar y realizar solicitudes de viáticos y transporte.

3. Como jefe de área debo poder visualizar y aprobar o no las solicitudes de viáticos y transportes de los emplead

no jefe de área financiera debo poder visualizar y aprobar o no las solicitudes de viáticos y transportes de los en

A realizar

Inicialización del proyecto de backend. Conexión a la base de datos. Codificación de los modelos y entidades de acceso a datos para la autenticación. Codificación del servicio de autenticación de usuarios. Pruebas del servicio de autenticación.
Inicialización del proyecto de frontend. Codificación de los componentes para las interfaces de autenticación. Maquetado de las interfaces de autenticación. Codificación del servicio encargado de realizar la autenticación con el backend.
Configuración del entorno de producción. Configuración de los aplicativos para su lanzamiento a producción. Despliegue del requerimiento.
Codificación de los modelos de acceso a datos de empleado y viáticos. Codificación del servicio de solicitud de viáticos. Aseguramiento y restricción el servicio de solicitud de viáticos a los usuarios con el rol de empleados
Codificación de los componentes para las interfaces de solicitud de viáticos. Maquetado de las interfaces de solicitud de viáticos Codificación del servicio encargado de realizar la solicitud de viáticos con el backend.
Despliegue del requerimiento.
Codificación de las entidades y modelos de acceso a datos del jefe de área. Codificación del servicio que permita recuperar solicitudes de viáticos en base a los empleados y al jefe de área as Aseguramiento y restricción el servicio de solicitud de viáticos a los usuarios con el rol de jefe de área Codificación del servicio que permita al jefe de área aprobar una solicitud.
Codificación de los componentes para las interfaces de visualización y aprobación de solicitudes. Maquetado de las interfaces de visualización y aprobación de solicitudes. Codificación del servicio encargado de recuperar las solicitudes y aprobarlas con el backend.
Despliegue del requerimiento.
Codificación de las entidades y modelos de acceso a datos de los usuarios de área financiera Codificación del servicio que permita recuperar solicitudes de viáticos de los empleados Aseguramiento y restricción el servicio de solicitud de viáticos a los usuarios con el rol de área financiera Codificación del servicio que permita al jefe de área financiera aprobar una solicitud.
Codificación de los componentes para las interfaces de visualización y aprobación de solicitudes. Maquetado de las interfaces de visualización y aprobación de solicitudes. Codificación del servicio encargado de recuperar las solicitudes y aprobarlas con el backend.
Despliegue del requerimiento.

Código de colores

Backend
Frontend
Despliegue

ESTÁNDARES GENERALES A UTILIZAR PARA EL PROYECTO

Estándares de desarrollo

En la presente actividad especificaremos los estándares de programación que utilizaremos en el desarrollo del proyecto, para ello abordaremos tanto los estándares generales como del lenguaje de programación JavaScript y la Base de Datos.

- Notación CamelCase en su tipo lowerCamelCase utilizada en el nombre de las variables, clases, métodos, funciones, entre otros aspectos clave del código fuente.
- Uso de nombres entendibles para nombrar las variables, no se hacen abreviaciones que no se entiendan.
- Las variables globales deberán de ser declaradas con un prefijo guion bajo o en su defecto indicar “global o gbl” al inicio del nombre de la variable (var _ejemplo o también var globalEjemplo).
- Las variables locales se declaran sin prefijo guion bajo como en el caso de las globales (let ejemplo).
- Para el tipo de variables booleanas, se debe usar el prefijo “es” (bool esValido, bool esInvalido).
- El nombre de las clases comienza con mayúscula (class Ejemplo).
- El nombre de los archivos deberá de ser idéntico a el nombre de la clase empleada o en su defecto podrá elegirse un nombre descriptivo que identifique fácilmente a todo el archivo.
- Para el uso de sangrías se deberá hacer mediante 1 TAB, 2 TAB, según sea la buena alineación del código y la identación.
- Los comentarios de una sola línea deberán usarse con “ // ”
- Los comentarios de más de una línea deberán usarse con “ /* */ “

  /*

  Comentarios

  Comentarios

  Comentarios

  */

- Los comentarios deberán estar al mismo nivel que el código.

  //Comentarios comentarios comentarios

  var ejemplo

  ejemplo = ejemplo + 1

- Las llaves { } deberán de utilizarse de la siguiente manera.

  If (x == y){

  //codigo

  }

- La ubicación de las variables globales deberá de ser en la parte superior del archivo.

- Se podrán declarar variables dentro de métodos o en otra parte del código siempre y cuando no se tengan que utilizar en otras partes y la ubicación de su declaración no cause conflicto en la ejecución del programa.

- Los métodos estarán entre 1 y 40 líneas de código, si hay más de 40 líneas se deberá reducir el código o en su defecto separar el método en dos diferentes.

- Cada método estará destinado a realizar una acción o tarea específica, se debe evitar la aplicación de multitareas en un mismo método.

- Se deberá evitar el uso de palabras reservadas del lenguaje de programación para las variables, métodos, funciones, clases, entre otros usos para los que no están destinadas, estas palabras reservadas las abordamos en la siguiente sección del documento.

Estándares de desarrollo para JavaScript

Para el desarrollo de este proyecto se ha decidido adoptar el estandar de codificación de Google para JavaScript, hemos optado por este estandar debido a la facilidad que existe para utilizarlo en el proyecto debido a la existencia de librerias de apoyo para buenas practicas de codificación que implementan este estandar.

- El nombre de los archivos debe ser de tipo lowercase, permitiendose el uso de guiones bajos o medios, excluyendose todo tipo de puntuación adicional.

- Clases, enumeraciones, funciones, constantes y otros simbolos que requieran utilizarse en otros modulos son exportados a través del objeto *exports*. Simbolos a exportar pueden declararse independientemente y exportados por separado o pueden ser definidos directamente en el objeto exports.

```
1.  const /** !Array<number> */ exportedArray = [1, 2, 3];
2.
3.  /** @return {number} */
4.  function exportedFunction() {
5.    return moduleLocalFunction() * 2;
6.  }
7.
8.  exports = {exportedArray, exportedFunction};
9.
10. /** @const {number} */
11. exports.CONSTANT_ONE = 1;
```

- Se debe usar la instrucción *import* para utilizar modulos de otros archivos, la extensión del archivo no es opcional, siempre debe incluirse .js, los nombres, los nombres de las importaciones deben usar la notación lowerCamelCase.

```
1.  import MyClass from '../my-class.js';
2.  import myFunction from '../my_function.js';
3.  import SOME_CONSTANT from '../someconstant.js';
```

- No se permite la creación de ciclos entre los modulos por medio de las instrucciones import y export, aunque la especificación ECMAScript lo permita. Un ciclo entre modulos se crea cuando se trata de importar o exportar algún módulo que ya exporte o importe al mismo módulo ya sea de manera directa o encadenadamente, a través de varias importaciones.

- La identación del codigo es a dos espacios.

- Solo se permiten 80 caracteres por linea, objetos, arreglos, clases, funciones y switchs se recomienda sean formateados siguiendo un estilo en bloque, dependiendo de la complejidad de las declaraciones.

```
const a = {
  a: 0,
  b: 1,
};
class Foo {
  constructor() {
    /** @type {number} */
    this.x = 42;
  }

/** @return {number} */
method() {
  return this.x;
}
}
some.reallyLongFunctionCall(arg1, arg2, arg3)
   .thatsWrapped()
   .then((result) => {
     // Indent the function body +2 relative to the indentation depth
     // of the '.then()' call.
     if (result) {
       result.use();
     }
   });
```

- Variables locales son declaradas con *const* o *let.* Se usa const por defecto a menos que una variable requiera ser reasignada.

- Debe evitarse el uso de las propiedades getters y setters, debido a que son potencialmente dificiles de razonar y tienen un soporte limitado en el compilador a menos que sean necesarios, por ejemplo al trabajar con algún framework como angular.

- Las funciones pueden contener definiciones de funciones anidadas. Si es util asignar un nombre a una función, debe asignarse a una variable const.

- Para funciones anidadas se recomienda el uso de la función flecha, se prefiere el uso de esta función sobre otros tipos. El lado izquierdo contiene los parametros, los parentesis del lado izquierdo de la función son opcionales su solo hay un parametro. El lado de la derecha contiene el cuerpo de la función.

```
class CallbackExample {
  constructor() {
    /** @private {number} */
    this.cachedValue_ = 0;

    getNullableValue((/** ?number */ result) => {
      this.cachedValue_ = result == null ? 0 : result;
    });
  }
}
```

- Las cadenas son delimitadas con comillas simples en lugar de comillas dobles.

- Para invocar al constructor de algún objeto, nunca debe omitirse los parentesis, debido a que puede generar errores.

```
new Foo();
```

- Para arreglos y enumeraciones siempre debe incluirse como al final de cada elemento.

```
const values = [
  'first value',
  'second value',
];
```

- El nombre de las variables sigue la notación lowerCamelCase, se permite hacer uso de otros simbolos cuando se utilice algún framework de desarrollo como angular. El nombre debe ser lo mas descriptivo posible sin abreviaciones ambiguas.

- El nombre de las enumeraciones sigue la notación UpperCamelCase.

- Para propiedades inmutables debe hacerse uso de constantes, una constante es una propiedad estática o una declaración module-local, pero debe hacerse incapié en que no todas las propiedades estáticas y module-local son constantes. Antes de elegir una propiedad como constante, debe considerarse si el campo realmente se siente como una constante profundamente inmutable. Por ejemplo, si cualquiera de los estados observables de esa instancia puede cambiar, es casi seguro que no es una constante. El nombre de las constantes debe ser completamente en mayusculas, con palabras separadas por guiones bajos.

```
1. // Constants
2. const NUMBER = 5;
3. /** @const */ exports.NAMES = ImmutableList.of('Ed', 'Ann');
4. /** @enum */ exports.SomeEnum = { ENUM_CONSTANT: 'value' };
5.
6. // Not constants
7. let letVariable = 'non-const';
8. class MyClass { constructor() { /** @const {string} */ this.nonStatic = 'non-
   static'; } };
9. /** @type {string} */ MyClass.staticButMutable = 'not @const, can be
   reassigned';
10. const /** Set<string> */ mutableCollection = new Set();
11. const /** ImmutableSet<SomeMutableType> */ mutableElements =
   ImmutableSet.of(mutable);
12. const Foo = goog.require('my.Foo');  // mirrors imported name
13. const logger = log.getLogger('loggers.are.not.immutable');
```

Estándares de desarrollo para Base de Datos

- Uso de nombres entendibles.

- Uso de SELECT *. No se debe de usar la sentencia “SELECT * FROM TABLA”, el traer todas las columnas de una tabla provoca un desbordamiento de IOPS en un servidor de base de datos, se debe de traer única y exclusivamente la información indispensable.

- No usar palabras reservadas de Mysql.

- Uso de Procedimientos almacenados para optimizar el sistema y el tiempo de respuesta en la conexión con la base de datos.

- Nombre de Stored Procedure. No guardar los procedimientos almacenados con un nombre con prefijo “sp_”.

- Usar la cláusula Join con estándar ANSI para juntar tablas.

- Utilizar SET NOCOUNT ON. Al crear procedimientos almacenados, para mejorar el desempeño de ADO.

- Las palabras reservadas deben ir en Mayúsculas.

- Ejecución de selección sentencia WHERE. Cuando se generen los criterios o condiciones de búsqueda en una sentencia SELECT dentro del WHERE se deberá de considerar que el orden de los filtros deberá de estar de izquierda a derecha ponderando los campos que abarquen un mayor universo a lo que abarquen menos.

- Estándares de codificación, comentarios, sangrías y buenas prácticas. El estándar para codificación, comentarios, sangrías, buenas prácticas visto en los apartados anteriores pueden ser aplicados también al estándar de base de datos según sea el caso.

VERSIONADOR DE CÓDIGO A UTILIZAR: GIT

Elegimos este versionado de código porque se ha posicionado como el estándar en la industria para versionadores de código lo que garantiza que la mayoría de los equipos tengan experiencia en su manejo a un buen nivel y suele ser el más usado en el mercado. De igual manera, proporciona una interfaz sencilla de usar y que tiene una curva de aprendizaje bastante pequeña para que los demás miembros puedan aprender rápidamente.

Enlace al repositorio:

https://github.com/manuelrojas19/Proyecto-Ingenieria-Software

SPRINTS

Número	FECHA INICIO	FECHA FIN	ACTIVIDADES*
SPRINT 1 TECNOLÓGICO	**06/05/2021**	**13/05/2021**	• Diseño de BD • Desarrollo del menú principal por perfiles • Construcción de BD • Codificación de los componentes para las interfaces de autenticación. • Conexión a la BD • Configuración de los aplicativos del segundo incremento para su lanzamiento a producción. • Maquetado HTML de las interfaces de solicitud de comisiones • Codificación del servicio encargado de realizar la solicitud de comisiones con el backend. • Maquetado de las interfaces de solicitud de comisiones.
SPRINT 2	**14/05/2021**	**27/05/2021**	• Aseguramiento y restricción del servicio de solicitud de viáticos a los usuarios con el rol de empleados. • Codificación de los modelos y entidades de acceso a datos para la autenticación. • Codificación de los modelos de acceso de datos de empleado y comisiones. • Codificación del servicio de solicitud de comisiones en el backend. • Inicialización del proyecto de backend • Inicialización del proyecto de Frontend. • Configuración del entorno de producción. • Acceso (Login) por perfil • Creación de procedimientos almacenados para insertar datos en todas las tablas • Creación de procedimientos almacenados para actualizar datos en todas las tablas • Codificación del servicio de autenticación de usuarios. • Codificación del servicio encargado de realizar la solicitud de comisiones en el backend. • Codificación de validaciones en el servicio de solicitud de comisiones • Codificación de los componentes para las interfaces de solicitud de comisiones. • Pruebas del servicio de autenticación • Creación de procedimientos almacenados para eliminar datos en todas las tablas

			• Despliegue del segundo incremento, solicitud de comisiones • Codificación de las entidades y modelos de acceso a datos de los comprobantes de comisión • Codificación del servicio que permita recuperar solicitudes de comisión a partir del departamento • Codificación del servicio que permita recuperar solicitudes de comisión a partir de los empleados. • Codificación del servicio que permita al administrador del área financiera aprobar una solicitud • Codificación del servicio que permita al jefe de área aprobar una solicitud. • Codificación del servicio que permita al empleado subir un comprobante de comisión • Codificación del servicio que permita recuperar comprobantes por comisión • Codificación del servicio encargado de recuperar las solicitudes y aprobarlas con el backend para los jefes de áreas. • Codificación del servicio encargado de recuperar las solicitudes y aprobarlas con el backend para los administradores del área financiera. • Codificación de los componentes para las interfaces de visualización y aprobación de solicitudes para los administradores del área financiera. • Codificación de los componentes para las interfaces de visualización y subida de comprobantes para los empleados • Codificación de los componentes para la visualización de reportes de comisiones asignadas por empleado • Codificación del servicio encargado de subir comisiones con el backend para los empleados. • Codificación de los componentes para las interfaces de visualización y aprobación de solicitudes para los jefes de área. • Pruebas del servicio de solicitud de comisiones.
SPRINT 3	**28/05/2021**	**17/06/2021**	• Codificación de los componentes para la visualización de reportes de gastos por empleado

			• Codificación del servicio que permita a los administradores del área financiera ajustar el presupuesto • Creación de procedimientos almacenados para los reportes.
SPRINT 4	**18/05/2021**	**22/06/2021**	• Actividades no terminadas el sprint anterior: • Codificación de los componentes para la visualización de reportes de gastos por empleado • Codificación del servicio que permita a los administradores del área financiera ajustar el presupuesto.

BACKLOG

Requerimientos	Prioridad
Que los empleados puedan registrar sus comprobantes desde el lugar de la comisión.	8
Que se realicen solicitudes de viáticos y de transporte.	1
Que las solicitudes pasen por un flujo de aprobación	2
Se debe llevar un control presupuestal por área.	
Los empleados deberán comprobar el 80% de sus gastos mediante facturas	3
El presupuesto deberá estar calendarizado y se dividirá por partida y por área.	7
El Sistema deberá permitir elaborar un reporte de actividades referente a la comisión.	6
Todas las solicitudes, aprobaciones y validaciones del Sistema se deberán realizar con una aprobación electrónica.	4
Generación de reportes: • Gastos desglosados por empleado * • Comisiones asignadas por empleado *	5

SPRINT	ACTIVIDADES	NO REALIZADAS	SPRINTS PASADOS
1	• Definir arquitectura • Diseño de BD • Desarrollo del menú principal por perfiles • Construcción de BD • Codificación de los componentes para las interfaces de autenticación. • Conexión a la BD • Configuración de los aplicativos del segundo incremento para su lanzamiento a producción. • Maquetado HTML de las interfaces de solicitud de comisiones • Codificación del servicio encargado de realizar la solicitud de comisiones con el backend. • Maquetado de las interfaces de solicitud de comisiones.		
2	• Aseguramiento y restricción del servicio de solicitud de viáticos a los		

	usuarios con el rol de empleados. • Codificación de los modelos y entidades de acceso a datos para la autenticación. • Codificación de los modelos de acceso de datos de empleado y comisiones. • Codificación del servicio de solicitud de comisiones en el backend. • Inicialización del proyecto de backend • Inicialización del proyecto de Frontend. • Configuración del entorno de producción. • Acceso (Login) por perfil • Creación de procedimientos almacenados para insertar datos en todas las tablas • Creación de procedimientos almacenados para actualizar datos en todas las tablas • Codificación del servicio de autenticación de usuarios. • Codificación del servicio encargado de		

	realizar la solicitud de comisiones en el backend. • Codificación de validaciones en el servicio de solicitud de comisiones • Codificación de los componentes para las interfaces de solicitud de comisiones. • Pruebas del servicio de autenticación • Creación de procedimientos almacenados para eliminar datos en todas las tablas • Despliegue del segundo incremento, solicitud de comisiones • Codificación de las entidades y modelos de acceso a datos de los comprobantes de comisión • Codificación del servicio que permita recuperar solicitudes de comisión a partir del departamento • Codificación del servicio que permita recuperar solicitudes de comisión a partir		

	de los empleados. • Codificación del servicio que permita al administrador del área financiera aprobar una solicitud • Codificación del servicio que permita al jefe de área aprobar una solicitud. • Codificación del servicio que permita al empleado subir un comprobante de comisión • Codificación del servicio que permita recuperar comprobantes por comisión • Codificación del servicio encargado de recuperar las solicitudes y aprobarlas con el backend para los jefes de áreas. • Codificación del servicio encargado de recuperar las solicitudes y aprobarlas con el backend para los administradores del área financiera. • Codificación de los componentes para las interfaces de visualización y		

	aprobación de solicitudes para los administradores del área financiera. • Codificación de los componentes para las interfaces de visualización y subida de comprobantes para los empleados • Codificación de los componentes para la visualización de reportes de comisiones asignadas por empleado • Codificación del servicio encargado de subir comisiones con el backend para los empleados. • Codificación de los componentes para las interfaces de visualización y aprobación de solicitudes para los jefes de área. • Pruebas del servicio de solicitud de comisiones.		
3	• Codificación de los componentes para la visualización de reportes de	• Codificación de los componentes para la visualización de reportes de	

	gastos por empleado • Codificación del servicio que permita a los administradores del área financiera ajustar el presupuesto • Creación de procedimientos almacenados para los reportes	gastos por empleado • Codificación del servicio que permita a los administradores del área financiera ajustar el presupuesto	
4			• Codificación de los componentes para la visualización de reportes de gastos por empleado • Codificación del servicio que permita a los administradores del área financiera ajustar el presupuesto

SCRUMBOARD

https://trello.com/invite/b/b3kKQkhp/dff3a5cdb083858dbe6ba3ca24e71b5b/proyecto-sistema-de-viaticos

Referencias **(Estándares)**

Google. (s.f). *Google JavaScript Style Guide.* https://google.github.io/styleguide/jsguide.html

EngineCore. (2019, 1 de octubre). MANUAL DE ESTANDARES DE CODIFICACIÓN Y BUENAS PRACTICAS EN C#, JAVASCRIPT, TRANSACT SQL Y ARQUITECTURA. http://www.enginecore.com.mx/assets/images/Estandar-Desarrollo-Plataforma-Engine-Core.pdf

Referencias (Arquitectura)

D. (2019, 30 octubre). *Estilo de arquitectura de microservicios - Azure Application Architecture Guide*. Microsoft Docs. https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices

EcuRed. (2019, 20 de junio). *Arquitectura de tres niveles*. https://www.ecured.cu/Arquitectura_de_tres_niveles

Referencias (versionador)

Nora Gonzalo Ciordia. (2021). 10 Comandos de Git Que Todo Desarrollador Debería Saber. 2021-05-12, de Free Code Camp Sitio web: https://www.freecodecamp.org/espanol/news/10-comandos-de-git-que-todo-desarrollador-deberia-saber/

1&1 IONOS Inc. (2021, 22 marzo). *Git vs. SVN: una comparativa del control de versiones*. IONOS Digitalguide. https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/git-vs-svn-una-comparativa-del-control-de-versiones/