Universidad Zaragoza

Trabajo Fin de Grado

Anexos

Plataforma integrada de pagos para la Universidad de Zaragoza

Payments integrated platform for the Zaragoza University

Autor

Samuel Gascón Gascón

Director y ponente

Pascual Pérez Sánchez (director)
Responsable técnico de la unidad de Administración Electrónica de la Universidad de Zaragoza

Joaquín Ezpeleta Mateo (ponente)
Departamento de Informática e Ingeniería de Sistemas

Escuela de Ingeniería y Arquitectura

2018

Índice

Anexo 1 - Instalación del framework Slim

Para la instalación del framework Slim, se van a seguir los siguientes pasos:

- En primer lugar, en el servidor de Apache proporcionado por la administración electrónica de la Universidad de Zaragoza, se va a ejecutar el comando *composer create-project slim/slim-skeleton pipuz.* Con este comando se crea un proyecto llamado *pipuz* con la siguiente distribución de carpetas para trabajar:

- logs: Para logs ante posibles errores o seguimientos. No se va a utilizar.

- public: Se encuentra el fichero index.php, que es el que levanta toda la API.

- src: Se pueden encontrar diferentes configuraciones. En esta aplicación, está implementado el middleware en el que se valida que exista un token de autenticación y que este sea correcto y no esté caducado.

- templates: No se va a usar, ya que sería la carpeta para dejar las plantillas de HTML, pero únicamente se va a usar Slim para el API REST.

- vendor: Aquí se almacenarán todas las dependencias que se van a instalar con composer.

Ahora, se va a crear una nueva carpeta llamada *app* y dentro de esta otras tres nuevas carpetas:

- route: Aquí irán los ficheros que se ejecutarán con las llamadas al API REST.

- model: Aquí irán los ficheros con los que se comunica la aplicación con la base de datos.

- lib: Aquí irán los ficheros *database.php,* en donde se creará la conexión con la base de datos con el framework PDO, y el fichero *reponse.php,* que será la clase que gestiona las respuestas de las llamadas a las bases de datos.

En está carpeta, también se creará el fichero *app_loader.php* que se utilizará para cargar todos los ficheros de cada una de las tres carpetas que se han creado anteriormente.

Anexo 2 – Base de datos

Tablas en la base de datos

- *pipuz_department*: Departamentos de la universidad a los que pertenecen las tasas. Atributos:

  - department_id: identificador del departamento.
  - name: nombre del departamento.

- *pipuz_user*: En esta tabla se definen los usuarios del sistema. Atributos:

  - nip: identificador del usuario en la Universidad.
  - name: nombre del usuario.
  - surname: apellidos del usuario.
  - dni: dni del usuario.
  - mail: correo electrónico del usuario.
  - admin: indica si es administrador (1) o no (0).

- *pipuz_admin*: En esta tabla se almacenan los diferentes usuarios que son administradores. Atributos:

  - nip: identificador del usuario que es administrador.
  - derpartment: identificador del departamento del que el usuario es administrador.
  - rol: el primer digito es siempre 1 y los siguientes indican con 1 ó 0 si tienen acceso a: ver los pagos del departamento, crear una tasa, actualizar una tasa, crear usuarios, crear un pago, actualizar pagos pagados con carta de pago y realizar acciones con descuentos (crear, borrar o actualizar).

- *pipuz_atrib_tasa*: Tabla en la que se almacenan los atributos que puede tener una tasa. Atributos:

  - atrib_tasa_id: identificador del atributo.
  - tipo: indica si es un atributo que siempre tiene el mismo valor (Único), puede ser un valor de un lista definida (Selección) o puede ser cualquier cosa (Variable).

- *pipuz_fijo*: Tabla en la que se almacena el único valor que puede tener un atributo único. Atributos:

  - fijo_id: identificador del atributo que es único.
  - value: valor que tiene el atributo único.

- *pipuz_seleccion*: Tabla en la que se almacenan todos los posibles valores que puede tener un atributo de selección. Atributos:

  - seleccion_id: identificador del atributo que es de selección.
  - pos_value: posible valor que puede tener el atributo de selección.

- *pipuz_atributo*: Tabla que únicamente sirve de información, en la que se define el nombre con el que aparecerá en la aplicación web un atributo. Atributos:

- atributo_id: identificador del atributo que aparece en la tabla de tasas y pagos.
- descripcion: nombre del atributo para que se muestren a los usuarios.

- *pipuz_tasa*: Esta tabla es la que representa las tasas disponibles para realizar pagos. Atributos:

- tasa_id: identificador de la tasa que es un autoincremental.
- name: nombre de la tasa.
- prize: precio normal de la tasa.
- department: identificador del departamento al que pertenece la tasa.
- description: descripción de la tasa.
- expire: máximo de días en los que se debe de pagar la tasa después de crearla
- a_academico: identificador de un atributo que representa al año académico de una tasa.
- a_contable: identificador de un atributo que representa al año contable de una tasa.
- convocatoria: identificador de un atributo que representa a la convocatoria de una tasa.
- experimentalidad: identificador de un atributo que representa la experimentalidad de una tasa.
- repeticion: identificador de un atributo que representa la repetición de una tasa.

- *pipuz_discount*: Tabla en la que se guardan los descuentos que se le pueden aplicar a los pagos de una tasa en particular. Atributos:

- descuento_id: identificador del descuento de una tasa.
- name: nombre del descuento.
- porcentaje: porcentaje de descuento que se hace del precio de la tasa a la que se aplica el descuento.
- tasa: identificador de la tasa a la que se aplica el descuento.

- *pipuz_task*: Representa los pagos de las tasas que realizan los usuarios del sistema. Atributos:

- referencia: identificador del pago que está definido por dos dígitos que pertenecen al año (ej. 18), seis dígitos que pertenecen al NIP del usuario al que pertenece el pago (ej. 683537) y cuatro dígitos a un incremental por usuario y año (ej. 0001).
- task_id: cuatro dígitos que pertenecen a un incremental por usuario y año.
- user: identificador del usuario al que pertenece el pago.
- year_task: dos dígitos que pertenecen al año.
- tasa: identificador de la tasa que se va a pagar en el pago.
- prize: precio del pago creado.
- create_date: fecha en la que se crea el pago.

- state: el estado del pago con valor 0 cuando está por pagar, 1 como pagado, 2 cancelado y 3 caducado.
- payment: identificador de la forma de pago en el caso de que el estado esté a 1.
- pay_date: fecha en la que se ha realizado el pago.
- expire_date: fecha en la que el pago va a caducar.
- discount: identificador del descuento de la tasa para el pago.
- a_academico: valor del atributo del año académico.
- a_contable: valor del atributo del año contable.
- convocatoria: valor del atributo de la convocatoria.
- experimentalidad: valor del atributo de la experimentalidad.
- repeticion: valor del atributo de repetición.

- *pipuz_payment*: En esta tabla se almacenan las diferentes formas de pago que existen en el sistema. Atributos:

- payment_id: identificador del método pago.
- type: tipo de pago que puede ser por Paypal, por TPV, al contado o por carta de pago.

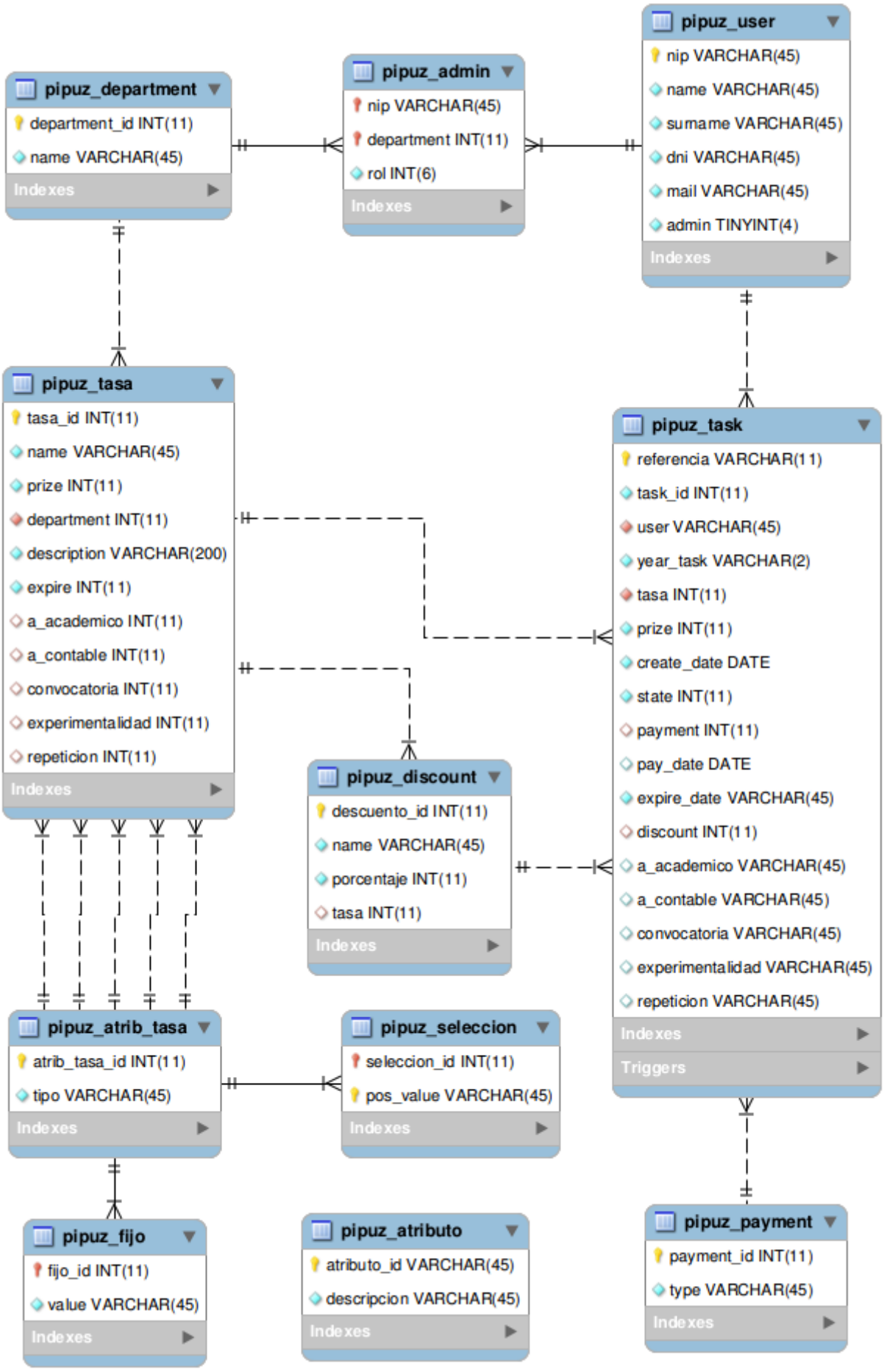


Figura 1: Esquema de la base de datos

Anexo 3 - Integración de la base de datos con PDO

La integración de la base de datos se ha realizado con el framework PDO de PHP. A través de este framework, se van a realizar las llamadas desde el API REST a la base de datos MySQL. Para la comunicación con la base de datos, se sigue el esquema de la figura 2, en donde se tienen diferentes módulos, que realizan llamadas a un módulo común y este realiza la llamada a la base de datos. Los diferentes módulos son: administración, departamento, formas de pago, tasa, tareas y usuarios.

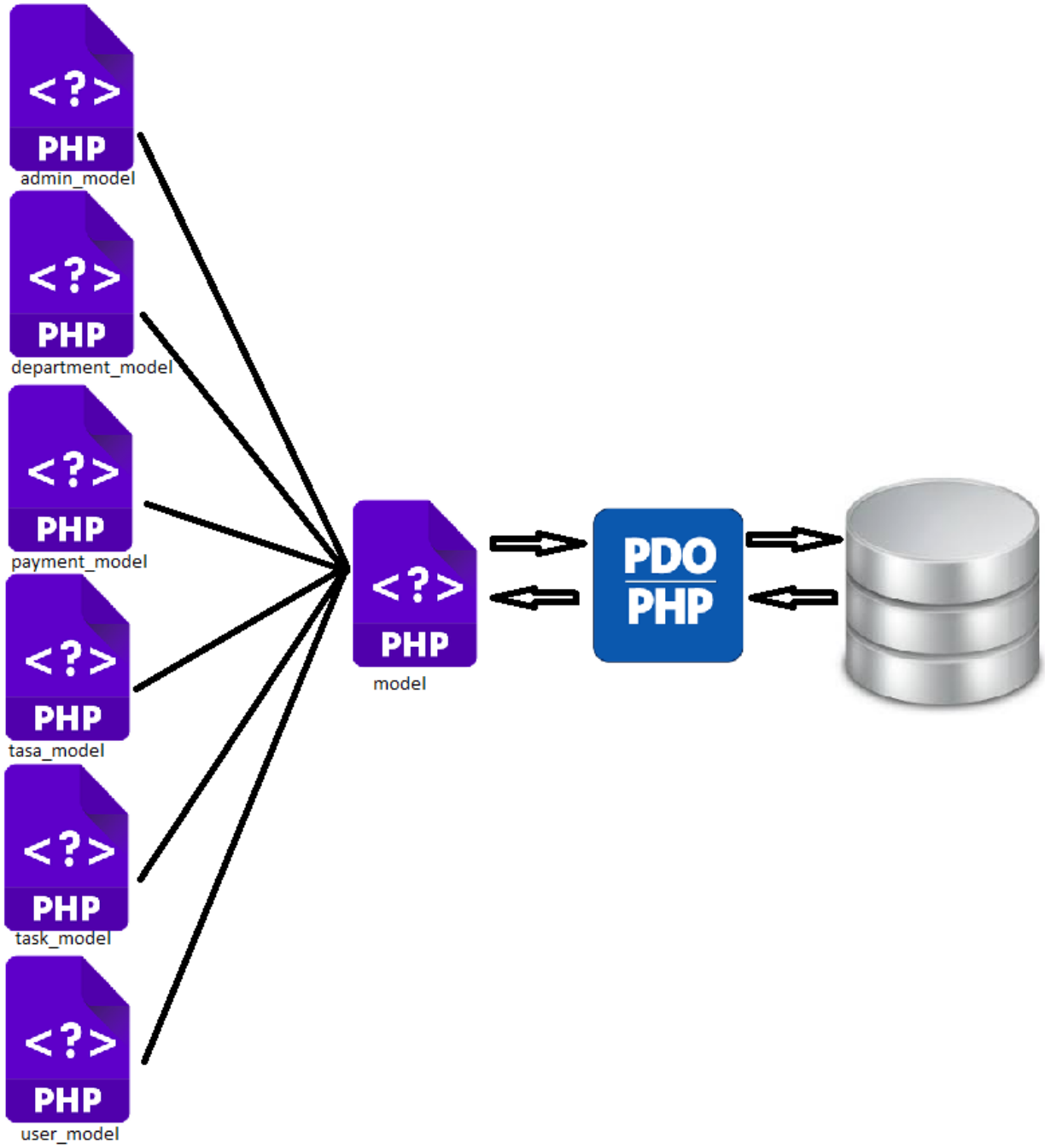


Figura 2: Esquema de la comunicación con la base de datos

Las diferentes llamadas de los diferentes módulos son las siguientes:

- admin_model:

- getTaskAdmin: Función que devuelve los pagos procedentes de un departamento en particular.
- getTasasAdmin: Función que devuelve las tasas procedentes de un departamento en particular.
- getDepartmentsAdmin: Función que devuelve los departamentos que son administrados por un usuario en particular.
- getColumns: Función que devuelve el nombre de todas las columnas de la tabla de tasas.
- addUnico: Función que crea un atributo único de una tasa. Primero crea el atributo en pipuz_atrib_tasa, después crea el atributo y su valor en pipuz_fijo para finalmente devolver el identificador del atributo creado.
- addVariable: Función que crea un atributo variable de una tasa. Primero crea el atributo en pipuz_atrib_tasa para finalmente devolver el identificador del atributo creado.
- addSeleccion: Función que crea un atributo de selección de una tasa. Primero crea el atributo en pipuz_atrib_tasa, después crea el atributo y sus posibles valores en pipuz_seleccion para finalmente devolver el identificador del atributo creado.
- getAtrib: Función que devuelve el tipo de un atributo indicando el identificador.
- getFijo: Función que devuelve el valor fijo de un atributo fijo indicando el identificador.
- getSelection: Función que devuelve los posibles valores de un atributo de selección indicando el identificador.
- getRoles: Función que devuelve los roles de administrador que tiene un usuario sobre un departamento.
- addAtributos: Función que añade el identificador de un atributo a la tasa correspondiente.
- createDiscount: Función en la que se crean descuentos, indicando la tasa, el nombre y el porcentaje.
- getDescuentos: Función que devuelve los descuentos que tiene una tasa.
- updateDescuentos: Función que actualiza un descuento de una tasa.
- deleteDescuentos: Función que elimina un descuento de una tasa.

- department_model:

- getDepartments: Función que devuelve un departamento en particular indicando su identificador o todos los departamentos.

- payment_model:

- getPayments: Función que devuelve todos los pagos disponibles en el sistema excepto el pago por efectivo.

- tasa_model:

- createTasa: Función que crea una tasa, en donde hay que indicarle todos los parametros necesarios: nombre (name), precio (prize), departamento (department), descripción (description) y caducidad (expire). Además también se puede añadir diferentes atributos (a_academico, a_contable, convocatoria, experimentalidad y repeticion).
- getTasa: Función que devuelve una tasa con todos sus campos indicando su identificador.
- updateTasa: Función que actualiza una tasa indicando su identificador y los campos que se quieren modificar.
- getTasas: Función que puede devolver todas las tasas en intervalos de 7 tasas o puede devolver todas las tasas del departamento que se le indique.
- getDepartmentName: Función que devuelve el nombre del departamento indicando su identificador.
- searchTasa: Función que devuelve las tasa que contengan en su nombre la cadena que se le indique.
- getNumTasas: Función que devuelve en número total de tasas que existe en el sistema.
- getDesc: Función que devuelve los descuentos que tiene una tasa.
- getAtributoDesc: Función que devuelve el nombre que tienen los atributos de tasas.

- task_model:

- createTask: Función que crea un pago, en donde hay que indicarle todos los parametros necesarios: usuario (user), tasa, precio (prize), fecha de creación (create_date), estado del pago (state), caducidad del pago (expire_date) y año (year_task). Además también se tienen que añadir los atributos atributos que contenga la tasa (a_academico, a_contable, convocatoria, experimentalidad o repeticion).
- getUserTask: Función que devuelve todas los pagos de un usuario y teniendo la opción de ponerle filtros de los diferentes campos de la tabla task.
- getTask: Función que devuelve un pago en particular indicando el usuario, el año y el identificador por usuario y año.

- getBestTask: Función que devuelve las 4 tasas que más se han utilizado en el sistema por todos los usuarios.
- payTask: Función que actualiza un pago a pagado.
- cancelarTask: Función que actualiza un pago a cancelado.
- comprobarCaducado: Comprueba la fecha de caducidad de todos los pagos de un usuario y, en el caso de que sea menor que la fecha actual, actualiza el pago a caducado.

- user_model:

- createUser: Función que crea un usuario, en donde hay que indicarle todos los parametros necesarios: NIP de la Universidad (nip), nombre (name), apellidos (surname), dni y correo electrónico (mail). El parametro de admin será siempre 0, porque el usuario no será administrador.
- getUser: Dado el NIP de la Universidad de un usuario, devuelve todos los datos del usuario.

Anexo 4 - API REST

Aqui se muestran todas las llamadas al API REST y se indica un breve descripción, los parámetros de entrada a la llamada y las posibles respuestas.

Las llamadas al módulo de login:

Tipo de llamada	Post
Nombre de la llamada	/api/loginUser
Parámetros	
userid (no nulo)	NIP del usuario de la Universidad
password (no nulo)	Contraseña administrativa del usuario de la Universidad
Descripción	
Esta llamada se utiliza para iniciar sesión en el sistema. Al dar un usuario y contraseña, que debe de ser el NIP de la Universidad y la contraseña administrativa, cifrado todo por TLS, se devolverá un token de autenticación. Este token deberá de ser enviado en cada llamada posterior que se realice y mediante un middleware se comprobará a qué usuario pertenece y si no tiene caducidad. Si el token no existe o no es válido, no se podrán realizar algunas llamadas (la mayoría). Para comprobar que el usuario y la contraseña son correctas, se realiza una comprobación contra el LDAP de la Universidad y se comprobará que el usuario esté registrado en el sistema.	
Respuestas	
Autenticación correcta	
code	Será el número 2.
token	Token de autenticación del usuario que deberá aparecer en todas las cabeceras de las llamadas posteriores.
admin	Indica 1 si el usuario es administrador y 0 si no lo es.
El usuario no existe en el sistema pero si en la universidad	
code	Será el número 0.
nip	Identificador de la universidad del usuario.
name	Nombre del usuario en el sistema de la universidad.
lastname	Apellidos del usuario en el sistema de la universidad.
nif	Número de identificación fiscal del usuario.
Error	

code	1
value	Indica cual es el error, desde que ha escrito mal la contraseña, a que se ha dejado algún campo por rellenar en la petición.

Tipo de llamada	Post
Nombre de la llamada	/api/newUser
Parametros	
userid (no nulo)	NIP del usuario de la Universidad
password (no nulo)	Contraseña administrativa del usuario de la Universidad
name (no nulo)	Nombre del usuario
lastname (no nulo)	Apellidos del usuario
dni (no nulo)	DNI del usuario
mail (no nulo)	Correo electrónico del usuario
Descripción	
Para registrar un usuario en el sistema se hace uso de esta llamada. El usuario que desee registrarse, deberá indicar su NIP de la Universidad, su contraseña administrativa, nombre, apellidos, DNI y correo electrónico. La comprobación del NIP, contraseña administrativa y DNI se realizará también contra el LDAP de la Universidad. Al realizarse siempre la comprobación del usuario y contraseña contra el LDAP, no será necesario almacenar la contraseña en la base de datos, pero si el usuario, para tener un control de sus pagos.	
Respuestas	
Registro correcto	
code	Será el número 0.
token	Token de autenticación del usuario que deberá aparecer en todas las cabeceras de las llamadas posteriores.
admin	Será un 0 ya que de primeras un usuario no se registra como administrador.
Error	
code	Será el número 1.
value	Indica cual es el error como que ha escrito mal la contraseña o el usuario, que el usuario ya está registrado en el sistema o que se ha dejado algún campo por rellenar en la petición.

Llamadas del módulo de usuario:

Tipo de llamada	Get
Nombre de la llamada	/api/getUser
Descripción	
Llamada que se utiliza para devolver el nombre del usuario del sistema. No necesita ningún parámetro ya que coge el identificador del usuario del token de autenticación.	
Respuestas	
Llamada correcta	
code	Será el número 0.
user	Nombre del usuario.
Error	
error	Es necesario autenticarse.

Tipo de llamada	Post	
Nombre de la llamada	/api/createTask	
Parametros		
tasa (no nulo)	Identificador de la tasa.	
atributos (array)	atributo	Nombre del atributo.
	tipo	Tipo del atributo que puede ser: fijo, variable o selección.
	valor	Valor del atributo.
Descripción		
Llamada para crear un pago. A esta llamada es necesario indicar la tasa que el usuario desea pagar y los atributos de la tasa en caso de que existan. En la base de datos se almacenará el identificador de la tasa, que será un autoincremental por usuario y año, el usuario, que se cogerá del token de autenticación, el año en dos dígitos, el identificador de la tasa, el precio, la fecha de creación, el estado en el que se encuentra el pago, la forma de pago que en el este momento no existirá, la fecha de caducidad del pago, si se le aplica algún descuento, los diferentes atributos que tenga y por último la referencia que está formado por el año, el identificador de usuario y el identificador de la tasa.		
Respuestas		
Tarea correcta		
code	Será el número 0.	
message	Tarea creada correctamente.	
Error		
code	Será el número 1.	

message	Mensaje del error.

Tipo de llamada	Get	
Nombre de la llamada	/api/allDepartments	
Parámetros		
id	Null o identificador de un departamento	
Descripción		
Llamada que devuelve o todos los departamentos o uno en concreto.		
Respuesta		
(array)	department_id	Identificador del departamento de la universidad.
	name	Nombre del departamento de la universidad.

Tipo de llamada	Get	
Nombre de la llamada	/api/allTasas	
Parámetros		
department	Null o identificador de un departamento	
num	Null o número de paginación que quieres mostrar, mostrando 7 tasas por paginación. Se usa para la paginación de la pantalla principal.	
Descripción		
Llamada que devuelve las tasas, indicando si quieres las tasas de algún departamento en particular.		
Respuesta		
(array)	tasa_id	Identificador del departamento de la universidad.
	name	Nombre del departamento de la universidad.
	department	Departamento al que pertenece la tasa.
	description	Descripción de la tasa.
	expire	Número de días en los

		que caduca la tasa una vez se cree el pago.	
	prize	Precio que tiene la tasa.	
	atributos (array)	atributo	Nombre del atributo.
		tipo	Tipo del atributo: Único, Selección o Variable.
		value(tipo es único)	Valor fijo del atributo.
		value(tipo es selección) (array)	Posible valor del atributo.
Tipo de llamada	Get		
Nombre de la llamada	/api/getTasa		
Parámetros			
tasa_id	Identificador de la tasa que se quiere mostrar.		
Descripción			
Llamada que devuelve todos los datos(nombre, descripción, precio, departamento, atributos...) de una tasa en particular indicando su identificador.			
Respuesta			
Existe la tasa indicada			
code	Será el número 0.		
tasa	tasa_id	Identificador de la tasa de la universidad.	
	name	Nombre de la tasa de la universidad.	
	department	Departamento al que pertenece la tasa.	
	description	Descripción de la tasa.	
	expire	Número de días en los	

		que caduca la tasa una vez se cree el pago.	
	prize	Precio que tiene la tasa.	
	atributos (array)	atributo	Nombre del atributo.
		tipo	Tipo del atributo: Único, Selección o Variable.
		value(tipo es único)	Valor fijo del atributo.
		value(tipo es selección) (array)	Posible valor del atributo.
Error			
code	Será el número 1.		
message	Mensaje de error, que indica que ha insertado ningún parametro o que la tasa indicada no existe.		

Tipo de llamada	Get
Nombre de la llamada	/api/numTasas
Descripción	
Llamada que devuelve el número de tasas del sistema.	
Respuesta	
(number)	Número de tasas que existe en el sistema

Tipo de llamada	Get	
Nombre de la llamada	/api/AllPayments	
Descripción		
Llamada que devuelve todas las posibles formas de pago que existen en el sistema.		
Respuesta		
(array)	payment_id	Identificador del método pago.
	type	Tipo de pago que puede ser por Paypal, por TPV, al contado o por carta de pago.

Tipo de llamada	Get	
Nombre de la llamada	/api/task	
Parámetros		
task_id	Los dos primeros dígitos son el año en el que se creó el pago y luego el identificador del pago por año y usuario.	
Descripción		
Devuelve un pago del usuario que realiza la llamada. El usuario lo coge del token de autenticación mientras que el pago se le indica con dos primeros dígitos del año en el que se creó el pago y el identificador del pago por año y usuario.		
Respuesta		
code	Será el número 0.	
task(array)	referencia	Identificador del pago, compuesto por el año (4 dígitos), el nip del usuario (6 dígitos) y el identificador del pago por usuario y año (3 dígitos).
	name	Nombre de la tasa del pago.
	department	Departamento al que pertenece la tasa del pago.
	description	Descripción de la tasa.
	prize	Precio del pago.

	state	Estado actual del pago.
	tasa	Identificador de la tasa del pago.
	task_id	Identificador del pago por usuario y año.
	expire_date	Fecha de caducidad de la tasa.
	create_date	Fecha en la que se ha creado la tasa.
atributos(array)	column	Descripción del atributo
	value	Valor del atributo
Error		
code	Será el número 1.	
message	No existe el pago indicado.	

Tipo de llamada	Get	
Nombre de la llamada	/api/history	
Parámetros		
option	Puede tener 4 valores: ‘null’ muestra todos los pagos de un usuario, 0 muestra los pagos por pagar, 1 muestra los pagados, 2 muestra los caducados y 3 los cancelados	
Descripción		
Esta llamada devuelve los pagos de un usuario pudiendo indicar el estado de los pagos que vas a recibir: por pagar, pagados, cancelados o caducados. También muestra el número de pagos que hay en cada estado.		
Respuesta		
code	Será el número 0.	
task	tasa_id	Identificador del departamento de la universidad.
	name	Nombre del departamento de la universidad.

	department	Departamento al que pertenece la tasa.
	description	Descripción de la tasa.
	expire	Número de días en los que caduca la tasa una vez se cree el pago.
	prize	Precio que tiene la tasa.
atributos(array)	column	Descripción del atributo
	value	Valor del atributo.
Error		
code	Será el número 1.	
message	No existe el estado indicado	

Tipo de llamada	Get	
Nombre de la llamada	/api/bestTask	
Descripción		
Llamada que devuelve las cuatro tasas más populares del sistema, es decir, las tasas que más veces se han creado.		
Respuesta		
(array)	tasa_id	Identificador de la tasa.
	name	Nombre de la tasa.

Tipo de llamada	Get	
Nombre de la llamada	/api/searchTasa	
Parámetros		
tasa	Cadena que debe pertenecer al nombre de alguna tasa.	
Descripción		
Se le pasa como parámetro una cadena y devuelve las tasas que tengan dicha cadena en su nombre.		
Respuesta		
(array)	tasa_id	Identificador del departamento de la

		universidad.
	name	Nombre del departamento de la universidad.
	department	Departamento al que pertenece la tasa.
	description	Descripción de la tasa.
	expire	Número de días en los que caduca la tasa una vez se cree el pago.
	prize	Precio que tiene la tasa.
	a_academico	Valor del atributo del año académico
	a_contable	Valor del atributo del año contable.
	convocatoria	Valor del atributo de la convocatoria.
	experimentalidad	Valor del atributo de la experimentalidad.
	repeticion	Valor del atributo de repetición.

Tipo de llamada	Get
Nombre de la llamada	/api/getCarta
Parámetros	
task_id	Los dos primeros dígitos son el año en el que se creó el pago y luego el identificador del pago por año y usuario.
Descripción	
Llamada que devuelve una carta de pago en PDF que contendrá los datos del usuario y los datos del pago que se desea realizar junto un código de barras y un código QR, para así, poder ir al banco a realizar el pago.	
Respuesta	
(PDF)	PDF que contendrá los datos del usuario y los datos del pago que se desea realizar junto un código de barras y un código QR, para así, poder ir al banco a realizar el pago.

Tipo de llamada	Post
Nombre de la llamada	/api/payPaypal
Parámetros	
task_id	Los dos primeros dígitos son el año en el que se creó el pago y luego el identificador del pago por año y usuario.
Descripción	
Esta llamada devuelve un link en el que se podrá realizar un pago mediante la plataforma Paypal.	
Respuesta	
Correcto	
code	Será el número 0.
link	Link para realizar el pago en Paypal.
Error de Paypal	
code	Será el número 1.
error	Descripción del error de Paypal.
Error del usuario	
code	Será el número 2.
error	*El pago no existe o ya se ha realizado.*

Tipo de llamada	Update
Nombre de la llamada	/api/paypalGood
Parámetros	
payment_id	Identificador del pago en Paypal.
PayerID	Identificador del pagador en Paypal.
Descripción	
Llamada en la que se comprueba que el pago se ha realizado correctamente sin sufrir ninguna modificación, se ejecuta el pago en paypal y se actualiza el estado del pago a *pagado* en la base de datos del sistema.	
Respuesta	
Correcto	
code	Será el número 0.
description	*El pago se ha realizado correctamente.*

task	Los dos primeros dígitos son el año en el que se creó el pago y luego el identificador del pago por año y usuario.	
Error de la base de datos		
code	Será el número 1.	
description	*No se ha completado el pago.*	
Error del pago		
code	Será el número 2.	
error	*El pago no existe o ya se ha realizado.*	
Tipo de llamada	Get	
Nombre de la llamada	/api/tpvData	
Parámetros		
task_id	Los dos primeros dígitos son el año en el que se creó el pago y luego el identificador del pago por año y usuario.	
Descripción		
Llamada que devuelve los datos necesarios para poder realizar el pago mediante el TPV.		
Respuesta		
Correcto		
code	Será el número 0.	
path	Link para realizar el pago en el TPV de Ceca.	
config	MerchantID	
	AcquirerBIN	
	TerminalID	
	TipoMoneda	
	Exponente	
	Cifrado	
	Pago_soportado	
	Idioma	
	Num_operacion	
	Importe	
	URL_OK	
	URL_NOK	
	Descripcion	
	Firma	
Error		

code	Será el número 1.
error	*El pago no existe o ya se ha realizado.*

Tipo de llamada	Update
Nombre de la llamada	/api/tpvGood
Parámetros	
Num_operacion	Los dos primeros dígitos son el año en el que se creó el pago y luego el identificador del pago por año y usuario.
Importe	Total del importe del pago multiplicado por 100.
Referencia	
Num_aut	
BIN	
FinalPAN	
Descripción	
Llamada en la que se comprueba que el pago se ha realizado correctamente sin sufrir ninguna modificación se actualiza el estado del pago a pagado en la base de datos del sistema y se devuelve la confirmación al TPV para realizar la ejecución del pago.	
Respuesta	
Correcto	
(cadena)	*$*$OKY$*$*
Error del sistema	
code	Será el número 1.
description	*No se ha completado el pago.*
Error del usuario	
code	Será el número 2.
description	*El pago no existe o ya se ha realizado.*

Tipo de llamada	Update
Nombre de la llamada	/api/cancelTask
Parámetros	
task	Los dos primeros dígitos son el año en el que se creó el pago y luego el identificador del pago por año y usuario.
Descripción	
Llamada para cancelar el pago de un usuario.	
Respuesta	
Correcto	
code	Será el número 0.
message	*Pago cancelado.*
Error	
code	Será el número 1.
description	*El pago no existe o ya se ha realizado.*

Tipo de llamada	Get
Nombre de la llamada	/api/isPay
Parámetros	
task_id	Los dos primeros dígitos son el año en el que se creó el pago y luego el identificador del pago por año y usuario.
Descripción	
Llamada que se realiza para comprobar si un pago se ha realizado correctamente.	
Respuesta	
Correcto	
code	Será el número 0.
task	Identificador del pago.
Error	
code	Será el número 1.
description	*El pago no se ha realizado correctamente.*

Tipo de llamada	Get	
Nombre de la llamada	/api/getDescuento	
Parámetros		
tasa_id	Identificador de la tasa de la que se quiere conocer el descuento.	
Descripción		
Llamada que se realiza para ver los descuentos que tiene una tasa.		
Respuesta		
Correcto		
code	Será el número 0.	
Descuentos (array)	descuento_id	Identificador del descuento.
	name	Nombre del descuento.
	porcentaje	Porcentaje que se descuenta del precio de la tasa.
	tasa	Identificador de la tasa.
Error		
code	Será el número 1.	
description	*Es necesario insertar todos los parámetros.*	

Llamadas al modulo de administración:

<table>
<tr><td>Tipo de llamada</td><td colspan="2">Get</td></tr>
<tr><td>Nombre de la llamada</td><td colspan="2">/api/getAdminDepartments</td></tr>
<tr><td colspan="3">Descripción</td></tr>
<tr><td colspan="3">Llamada que devuelve los departamentos que un usuario tiene acceso al módulo de administración y los roles que tiene en dicho departamento.</td></tr>
<tr><td colspan="3">Respuesta</td></tr>
<tr><td colspan="3">Correcto</td></tr>
<tr><td rowspan="4">department (array)</td><td>name</td><td>Nombre del departamento del que el usuario es administrador.</td></tr>
<tr><td>department_id</td><td>Identificador del departamento del que el usuario es administrador.</td></tr>
<tr><td>rol</td><td>Rol en cadena (ej. 111101011) con el primer dígito de control que tiene el usuario en el departamento.</td></tr>
<tr><td>Roles (array)</td><td>Rol en array (ej. [1,1,1,1,0,1,0,1,1]) con el primer dígito de control que tiene el usuario en el departamento.</td></tr>
<tr><td>rol (array)</td><td colspan="2">Rol en array (ej. [1,1,1,0,1,0,1,1]) que tiene el usuario.</td></tr>
</table>

Tipo de llamada	Get	
Nombre de la llamada	/api/getAdminTasks	
Parámetros		
department	Identificador del departamento.	
order	Cadena que indica *campo de ordenacion* + *ASC/DESC* (*ASC* indica orden ascendente y *DESC* indica orden descendente).	
filtro	Cadena en donde se busca en el nombre, descripción o usuario de la tasa si existe alguna coincidencia.	
Descripción		
Llamada que devuelve todos los pagos realizados en un departamento, con la posibilidad de ordenar por un campo en específico y de filtrar por nombre. La llamada solo se podrá realizar si el usuario es administrador de ese departamento y tiene el rol correspondiente.		
Respuesta		
Correcto		
(array)	referencia	Identificador del pago.
	task_id	Identificador del pago por usuario y año.
	user	Usuario al que pertenece el pago.
	year_task	Año en el que se ha creado el pago (2 dígitos).
	tasa	Identificador de la tasa.
	prize	Precio que tiene la tasa.
	create_date	Fecha en la que se ha creado el pago
	state	Estado actual del pago.
	payment	Forma de pago que se ha realizado.
	pay_date	Fecha en la que se ha realizado el pago.
	expire_date	Fecha en la que caduca la tarea.

	discount	Descuento que se aplica al pago.
	a_academico	Valor del atributo del año académico
	a_contable	Valor del atributo del año contable.
	convocatoria	Valor del atributo de la convocatoria.
	experimentalidad	Valor del atributo de la experimentalidad.
	repeticion	Valor del atributo de repetición.
	estado	Estado del pago: *Pagado, Cancelado, Caducado, Sin pagar.*

Tipo de llamada	Get	
Nombre de la llamada	/api/getAdminTasas	
Parámetros		
department	Identificador del departamento.	
Descripción		
Devuelve todas las tasas con todos los datos de un departamento. La llamada solo se podrá realizar si el usuario es administrador de ese departamento y tiene el rol correspondiente.		
Respuesta		
Correcto		
(array)	tasa_id	Identificador de la tasa.
	name	Nombre del departamento de la universidad.
	department	Departamento al que pertenece la tasa.
	description	Descripción de la tasa.
	expire	Número de días en los que caduca la tasa una vez se cree el

		pago.
	prize	Precio de la tasa.
	a_academico	identificador de un atributo que representa al año académico
	a_contable	identificador de un atributo que representa al año contable.
	convocatoria	identificador de un atributo que representa a la convocatoria.
	experimentalidad	identificador de un atributo que representa a la experimentalidad.
	repeticion	Identificador de un atributo que representa a la repetición.

Tipo de llamada	Post	
Nombre de la llamada	/api/createTasa	
Parámetros		
name	Nombre del departamento de la universidad.	
department	Departamento al que pertenece la tasa.	
description	Descripción de la tasa.	
expire	Número de días en los que caduca la tasa una vez se cree el pago.	
prize	Precio que tiene la tasa.	
options	atributo	Nombre del atributo.
	tipo	Tipo del atributo: Único, Selección o Variable.
	value(tipo es único)	Valor fijo del atributo.
	value(tipo es	Posible valor del

	selección) (array)	atributo.
Descripción		
Llamada que crea una tasa de un departamento indicando todos sus datos y atributos si los tiene. La llamada solo se podrá realizar si el usuario es administrador de ese departamento y tiene el rol correspondiente.		
Respuesta		
Correcto		
code	Será el número 0.	
message	Tasa creada correctamente.	
tasa	Identificador de la tasa creada.	
Error		
code	Será el número 1.	
message	Mensaje del error	

Tipo de llamada	Get		
Nombre de la llamada	/api/showAtrib		
Parámetros			
tasa_id	Identificador de la tasa		
Descripción			
Muestra todos los atributos que tiene una tasa y sus características. La llamada solo se podrá realizar si el usuario es administrador de ese departamento y tiene el rol correspondiente.			
Respuesta			
Correcto			
code	Será el número 0.		
fijos	prize	Precio de la tasa.	
	description	Descripción de la tasa.	
	expire	Días de caducidad de la tasa desde su creación.	
atributos	opt	Nombre del atributo.	
	valor	value	0,1 ó 2

			dependiendo si es variable, único o de selección.
		desc	Tipo de atributo: *Variable*, *Único* o *Selección*
	value	Valor del atributo en caso de ser fijo	
	selections (array)	num	Número del posible valor de la selección
		sel	Posible valor de la selección.
	num	Número de atributo de la tasa.	
	numSel	Número total de posibles valores en caso de *Selección.*	
Error de parámetros			
code	Será el número 1.		
message	*Es necesario insertar todos los parámetros obligatorios.*		
Error de permisos			
code	Será el número 2.		
message	*No tiene permisos para realizar esta acción.*		

Tipo de llamada	Update		
Nombre de la llamada	/api/updateTasa		
Parámetros			
id	Identificador de la tasa.		
atributos (array)	opt	Tipo de atributo.	
	valor	desc	Tipo de valor:

			Único, Selección, Variable.
	value (valor *Único*)	Valor en caso de que el atributo sea fijo	
	selections (valor *Selección*)	sel	Posible valor de la selección.
updates (array)	opt	Tipo de atributo.	
	valor	0,1 ó 2 dependiendo si es variable, único o de selección.	
	value (valor *Único*)	Valor en caso de que el atributo sea fijo	
	selections (valor *Selección*)	sel	Posible valor de la selección.
Descripción			
Actualiza los datos de un tasa, pudiendo modificar los atributos y añadir nuevos. La llamada solo se podrá realizar si el usuario es administrador de ese departamento y tiene el rol correspondiente.			
Respuesta			
Correcto			
(cadena)	*Modificado -tasa_id-*		
Error			
code	Será el número 1.		
message	Mensaje del error		

Tipo de llamada	Post	
Nombre de la llamada	/api/createDescuento	
Parámetros		
discount (array)	name	Nombre del descuento.
	porcentaje	Porcentaje de descuento.
department	Identificador del departamento de la tasa.	
tasa	Identificador de la tasa.	
Descripción		
Crear un descuento para una tasa en particular, indicando el nombre y el porcentaje de descuento. La llamada solo se podrá realizar si el usuario es administrador de ese departamento y tiene el rol correspondiente.		
Respuesta		
Correcto		
code	Será el número 0.	
message	*Descuento creado correctamente.*	
Error		
code	Será el número 1.	
message	Mensaje del error	

Tipo de llamada	Update	
Nombre de la llamada	/api/updateDescuento	
Parámetros		
updateDiscount (array)	descuento_id	Identificador del descuento.
	porcentaje	Nuevo porcentaje de descuento.
Descripción		
Modifica el porcentaje de un descuento en una tasa. La llamada solo se podrá realizar si el usuario es administrador de ese departamento y tiene el rol correspondiente.		
Respuesta		
Correcto		

code	Será el número 0.
message	*Actualización con éxito.*
Error	
code	Será el número 1.
message	*No ha actualizado ningún descuento.*

Tipo de llamada	Delete	
Nombre de la llamada	/api/deleteDescuento	
Parámetros		
deleteDiscount (array)	descuento_id	Identificador del descuento.
Descripción		
Elimina los descuentos que se le pasen por parámetro.		
Respuesta		
Correcto		
code	Será el número 0.	
message	*Descuentos borrados correctamente*	
Error		
code	Será el número 1.	
message	*No ha borrado ningún descuento.*	

Tipo de llamada	Get
Nombre de la llamada	/api/getColumns
Parámetros	
nip	NIP del usuario en la Universidad.
Descripción	
Dado un NIP, devuelve el nombre y apellidos de un usuario. La llamada solo se podrá realizar si el usuario es administrador y tiene el rol correspondiente.	
Respuesta	
Existe el usuario	
code	Será el número 1.

name	Nombre del usuario.
surname	Apellidos del usuario.
No existe el usuario	
code	Será el número 0.

Tipo de llamada	Post
Nombre de la llamada	/api/createAdminTask
Parámetros	
user	NIP del usuario en la Universidad.
tasa	Identificador de la tasa.
paid	0 si el usuario lo pagará en su sistema y 1 si el usuario lo paga al contado.
Descripción	
Crea un pago para un usuario. Se puede indicar si el pago es al contado o no. La llamada solo se podrá realizar si el usuario es administrador del departamento de la tasa en la que crea el pago y tiene el rol correspondiente.	
Respuesta	
Existe el usuario	
code	Será el número 0.
message	Mensaje de operación correcta.
Error	
code	Será el número 1.
message	Mensaje de error.

Tipo de llamada	Update	
Nombre de la llamada	/api/payWithLetter	
Parámetros		
tasks	user	NIP del usuario en la Universidad.
	task_id	Identificador de la tasa.
	date	Fecha en la que se ha realizado el pago.
Descripción		
Llamada que se utiliza para actualizar los pagos que se han realizado por carta de pago en el banco. La llamada solo se podrá realizar si el usuario es administrador de ese departamento y tiene el rol correspondiente.		
Respuesta		
Todo realizado correctamente		
code	Será el número 0.	
message	*Se han actualizado todos los pagos*.	
errors	Será un array vacío.	
Error		
code	Será el número 1.	
message	Mensaje de error.	
errors (array)	task_id	Pago que ha tenido el error.
	user	Usuario que ha tenido el error.
	message	Mensaje de error.
Faltan datos		
code	Será el número 2.	
message	*Inserte todos los datos.*	

Anexo 5 - Carta de Pago

Para la creación de las cartas de pago, hay que tener en cuenta tres aspectos importantes para su implementación: código de barras, código *QR* y generar un archivo *PDF* a través de un texto en *HTML.*

Código de barras

Para poder realizar la generación del código de barras, hay que seguir los siguientes pasos:

- Instalar la librería *Php Barcode Generator* con el comando *composer require picqer\php-barcode-generator.*

- En el archivo *PHP* en el que se vaya a generar el código (será *user.php*), indicar que se va a utilizar dicha librería con el comando *use \Picqer\Barcode.*

- Por último, para la generación del código, se realiza la llamada *getBarcode(parametros).*

Código de *QR*

Para poder realizar la generación del código *QR,* hay que seguir los siguientes pasos:

- Descargar la librería *PHP QR Code.*

- Dejar la carpeta descargada en la carpeta donde permanecen los archivos *PHP* donde están las llamadas al *API REST* (app/app/route).

- En el archivo *PHP* en el que se vaya a generar el código (será *user.php*), se importa el archivo *qrlib.php*, que está en la carpeta que se ha descargado.

- Por último, para la generación del código, se realiza la llamada *Qrcode::image(parametros).*

Generación de archivo *PDF*

Para poder generar el archivo *PDF*, hay que seguir los siguientes pasos:

- Descargar la librería *Dompdf.*

- Dejar la carpeta descargada en la carpeta donde permanecen los archivos *PHP* donde están las llamadas al *API REST* (app/app/route).

- En el archivo *PHP* en el que se vaya a generar el código (será *user.php*), se importa el archivo *autoload.inc.php*, que está en la carpeta que se ha descargado.

- A continuación, se debe de escribir el *PDF* que queremos generar en formato *HTML.*

- Por último, se llama a la función *loadHtml(html)*, para carga el *HTML* que se ha escrito, y la función *render()*, para generar el *PDF.*

Anexo 6 – Instalación de las pasarelas de pago

Instalación del TPV de CECA

Para la instalación del TPV de CECA en el sistema, se van a seguir los siguientes pasos:

- Lo primero será tener una cuenta en CECA. Como la Administración Electrónica de la Universidad de Zaragoza ya tiene una, se va a utilizar esa cuenta.

- A continuación se accederá a la consola del TPV en la página https://comercios.ceca.es/pruebas. El desarrollo se va a hacer en el entorno de pruebas. Se accederá a la configuración y se establecera la URL de confirmación del pago a https://nuez.unizar.es/~pipuz/app.php/api/tpvGood, que será la dirección en la que se confirma el pago en el API REST. También se indicará que se requerirá la respuesta OK.

- Para hacer funcionar el TPV en el sistema, será necesario tener la librería del TPV de CECA que se instalará con el comando *composer require ceca/tpv*.

- A continuación, habrá que modificar el fichero de configuración *config.php.* Este fichero estará en la ruta */vendor/ceca/tpv*. El fichero tiene un array con los siguientes elementos: *Environment, MerchantID, AcquireBIN, TerminalID, ClaveCifrado, TipoMoneda, Exponente, Cifrado, Idioma* y *Pago_Soportado.*

- La manera que tendrá el usuario al TPV, será a través de un formulario HTML. Cuando el usuario quiera pagar por TPV, se generará una llamada GET al API REST con la ruta */api/tpvData* que dará como resultado un el formulario HTML.

- Una vez dados los datos al TPV de CECA, este enviará una llamada al API REST a la ruta *api/tpvGood,* en donde se comprobará que el pago es correcto y se devolverá al TPV una confirmación con la cadena *$*$OKY$*$.*

Instalación del TPV de Paypal

Para la instalacion de Paypal en el sistema se van a seguir los siguientes pasos:

- Lo primero será tener una cuenta personal en Paypal. Una vez creada la cuenta, se deberá de acceder a *Sandbox,* y que se va a realizar el desarrollo en un entorno de prueba, y crear una *cuenta personal* y una *cuenta business.*

- Una vez creada la *cuenta business*, se deberá de crear una aplicación, indicando el nombre de la aplicación y la *cuenta business*. Una vez creada, se habrán creado dos claves: *Client ID* y *Secret.* Habrá que guardarlas para utilizarlas posteriormente.

- A continuación, habrá que instalar la librería de Paypal en PHP en nuestro API REST con el comando *composer require paypal/rest-api-sdk-php:**.

- Una vez realizados todos los pasos anteriores, ya se pueden implementar los pagos con Paypal en el API REST. Para poder acceder a la aplicación creada, habrá que utilizar la clase *ApiContext* de la librería de Paypal. Para acceder, será necesario utilizar las dos claves generadas anteriormente. Para crear el pago, habrá que indicar el método de pago (paypal), el total de la transacción, la moneda (euro), y una descripción, que en la aplicación será el identificador de la tasa por pagar, el nombre de la tasa y el identificador del pago en el

sistema. A continuación, se indicará la direccióne a la que hay que redirigir al usuario en caso de haber efectuado el pago correcto (que será https://nuez.unizar.es/~pipuz/paypalGood) y el pago incorrecto (que será https://nuez.unizar.es/~pipuz/paypalBad). Por último, se devolverá al usuario el link de Paypal para efectuar el pago.

- Una vez efectuado el pago por parte del usuario en la página de Paypal, hará falta una confirmación desde el sistema. Para ello se volverá a acceder a la aplicación como se ha hecho anteriormente, se validará que el pago que se ha hecho es correcto, y se dará una confirmación a Paypal de la ejecución del pago.

Anexo 7 - AngularJS

Para la interacción del API REST con la interfaz web, se va a utilizar el framework de JavaScript, AngularJS.

El API de AngularJS se basará en crear un módulo maestro y diferentes controladores para los diferentes módulos de la aplicación web. Los diferentes controladores son: *AdminController, HomeController, LoginController y TaskController.*

En primer lugar, se va a crear el módulo maestro llamado *master.* En el módulo se van a configurar a que html se accederá según la ruta a la que se dirija el usuario, se va a permitir html5 y se va a configurar un interceptor, que meta siempre el token de autorización en la cabecera de las llamadas *HTTP*. Por último, se crearan unas funciones que serán accesibles desde cualquier sitio que pertenezca al módulo maestro, que son:

- closeSesion: Borra el token de autorización y redirige al usuario a la pantalla principal.

- searchTasa: Realiza una llamada *HTTP* al API REST a la ruta */api/searchTasa/* para que busque una tasa cuyo nombre contenga la cadena que se le pasa por parametro.

- createTaskClicked: Asigna a una variable la tasa que se le pasa por parametro y redirige a la pantalla de crear una tasa.

- getUser: Realiza una llamada *HTTP* al API REST a la ruta */api/getUser* para que busque al usuario propietario del token de autorización y mostrar su nombre y apellidos.

A continuación se van a explicar los diferentes controladores que se han nombrado anteriormente.

En primer lugar está el controlador *AdminController,* que es el encargado de realizar las funciones para los usuarios administradores. Las diferentes funciones de este controlador son:

- loadAdminDepartments: Realiza una llamada *HTTP* al API REST a la ruta */api/getAdminDepartments* para conocer los departamentos de los que el usuario es administrador. También se comprueba los roles que tiene el usuario y finalmente se realiza una llamada *HTTP* al API REST a la ruta */api/getAdminTasks* pasándole como parámetro el primer departamento devuelto para conocer los pagos de dicho departamento.

- loadAdminTasks: Realiza una llamada *HTTP* al API REST a la ruta */api/getAdminTasks* pasándole como parámetro un departamento para que devuelvo los pagos de ese departamento.

- changeDepartment: Cambia el departamento para el rol *Crear Tasa.*

- changeDepartment2: Cambia el departamento para el rol *Actualizar Tasa.*

- order: Realiza una llamada *HTTP* al API REST a la ruta */api/getAdminTasks* indicandole el orden en el que quieres que se muestren los diferentes pagos de un departamento.

- filtro: Realiza una llamada *HTTP* al API REST a la ruta */api/getAdminTasks* indicandole una cadena que quieres que aparezca en el usuario del pago, descripción o nombre de la tasa del pago de un departamento.

- changeValue: Cambia la pestaña que aparece en el módulo de administración. En caso de que la pestaña sea *Actualizar Tasa, Generar recibo, Pagar al contado* o *Descuento,* realiza una

llamada *HTTP* al API REST a la ruta */api/getAdminTasks* pasándole como parámetro un departamento para que devuelva los pagos de ese departamento. En caso de que sea *Crear Tasa,* realiza una llamada *HTTP* al API REST a la ruta */api/getColumns* para cargar el nombre de los atributos que tienen las tasas.

- loadAdminTasasRec: Cambia el departamento para el rol *Generar Recibo* y realiza una llamada *HTTP* al API REST a la ruta */api/getAdminTasks* pasándole como parámetro el departamento que se ha elegido en la pestaña *Generar Recibo.*

- loadAdminTasas: Cambia el departamento para el rol *Pagar al contado* y realiza una llamada *HTTP* al API REST a la ruta */api/getAdminTasks* pasándole como parámetro el departamento que se ha elegido en la pestaña *Pagar al contado.*

- newTasa: Realiza una llamada *HTTP* al API REST a la ruta */api/createTasa* pasándole como parámetros los campos necesarios de una tasa: nombre, descripción, precio, caducidad, departamento y los diferentes atributos que pueda tener.

- updateTasa: Realiza una llamada *HTTP* al API REST a la ruta */api/updateTasa* pasándole como parámetros los cambios realizados sobre los atributos, los nuevos atributos de la tasa, los campos que cambian de la tasa y el identificador de la tasa.

- newTask: Realiza una llamada *HTTP* al API REST a la ruta */api/createAdminTask* para crear un pago, indicando el usuario del pago, el identificador de la tasa y que el estado del pago sea sin pagar, para así generar el recibo.

- newPay: Realiza una llamada *HTTP* al API REST a la ruta */api/createAdminTask* para crear un pago, indicando el usuario del pago, el identificador de la tasa y que el estado del pago sea pagado, ya que se realiza el pago al contado.

- addOption: Añade un nuevo registro en un array para los atributos en la creación de una tasa. Los parametros del array serán: nombre del atributo (*opt),* de que tipo es el valor del atributo (si es fijo, variable o de selección) (*valor),* valor del atributo en caso de ser fijo (*value)*, array con los posibles valores en caso de ser un atributo de selección (*selections),* posición del registro en el array (*num)* y el número de posibles valores que tiene en caso de ser un atributo de selección (*numSel).* Este registro estará vacío pero se indicará su posición en el array.

- deleteOption: Elimina un registro del array de atributos para la creación de una tasa indicando la posición del registro en el array.

- addSelection: Añade un nuevo registro al parámetro *selections* del array de atributos para la creación de una tasa indicando el número de su posición. El parámetro *selections* es un array con los siguientes parámetros: posición del registro en el array *selections* (*num*) y posible valor (*sel*). La posición se coge del parámetro del array de atributos *numSel.*

- deleteSelection: Elimina un registro del parámetro *selections* del array de atributos para la creación de una tasa, indicando la posición registro en el array y en el parametro *selections.*

- addUpdateOp: Añade un nuevo registro en un array para los atributos en la actualización de una tasa. Los parametros del array serán: nombre del atributo (*opt),* de que tipo es el valor del atributo (si es fijo, variable o de selección) (*valor),* valor del atributo en caso de ser fijo (*value)*, array con los posibles valores en caso de ser un atributo de selección (*selections),* posición del registro en el array (*num)* y el número de posibles valores que tiene en caso de ser un atributo de selección (*numSel).* Este registro estará vacío pero se indicará su posición en el array.

- deleteUpdateOp: Elimina un registro del array de atributos para la actualización de una tasa indicando la posición del registro en el array.

-deleteAtrib: Elimina un registro del array de atributos ya existentes en la actualización de una tasa indicando la posición del registro en el array.

- addUpdateSel: Añade un nuevo registro al parámetro *selections* del array de atributos para la actualización de una tasa indicando el número de su posición. El parámetro *selections* es un array con los siguientes parámetros: posición del registro en el array *selections* (*num*) y posible valor (*sel*). La posición se coge del parámetro del array de atributos *numSel.*

- addAtribSel: Añade un nuevo registro al parámetro *selections* del array de atributos ya existentes en la actualización de una tasa indicando el número de su posición. El parámetro *selections* es un array con los siguientes parámetros: posición del registro en el array *selections* (*num*) y posible valor (*sel*). La posición se coge del parámetro del array de atributos *numSel.*

- deleteUpdateSel: Elimina un registro del parámetro *selections* del array de atributos para la actualización de una tasa, indicando la posición registro en el array y en el parametro *selections.*

- deleteAtribSel: Elimina un registro del parámetro *selections* del array de atributos ya existentes en la actualización de una tasa, indicando la posición registro en el array y en el parametro *selections.*

- getUser: Realiza una llamada *HTTP* al API REST a la ruta */api/getuUser* pasándole como parámetro el DNI de un usuario y muestra el nombre y apellidos de dicho usuario.

- addPay*:* Añade un nuevo registro en un array para los pagos que se han realizado con carta de pagos en el banco. Los parametros del array serán: identificador del pago (*task_id),* departamento de la tasa (department*),* DNI del usuario que ha realizado el pago (*user)*, fecha en la que se realizó el pago (*date)* y posición del registro en el array (*num).* Este registro estará vacío pero se indicará su posición en el array.

- deletePay: Elimina un registro del array para los pagos que se han realizado con carta de pagos en el banco, indicando la posición registro en el array.

- payDepartment: Cambia el departamento para el rol *Actualizar Carta de Pago.*

- updatePays: Realiza una llamada *HTTP* al API REST a la ruta */api/payWithLetter,* pasando como parámetro el array utilizado en las funciones *addPay* y *deletePay,* y actualiza a pagados los pagos de ese array.

- showAtrib: Realiza una llamada *HTTP* al API REST a la ruta */api/showAtrib,* pasando como parámetro el identificador de la tasa que se va a actualizar, devolviendo los valores de la tasa y sus atributos. A continuación, realiza una llamada *HTTP* al API REST a la ruta */api/getColumns* para cargar el nombre de los atributos que tienen las tasas.

- addDiscount: Añade un nuevo registro en un array para la creación de descuentos. Los parametros del array serán: nombre (*name),* porcentaje (*porcentaje)* y posición del registro en el array (*num).* Este registro estará vacío pero se indicará su posición en el array.

- deleteAddDiscount: Elimina un registro del array para la creación de descuentos, indicando la posición registro en el array.

- createDiscount: Realiza una llamada *HTTP* al API REST a la ruta */api/createDescuento,* pasando como parámetro el array utilizado en las funciones *addDiscount* y *deleteDiscount,* el departamento de los pagos y el identificador de la tasa, para que se cree en dicha tasa los descuentos indicados.

- descuento: Cambia el valor de la acción que se realizará sobre descuento: 0 será crear, 1 actualizar y 2 borrar.

- desc: Realiza una llamada *HTTP* al API REST a la ruta */api/getDescuento,* pasando como parámetro una tasa, y devuelve los descuentos que tiene dicha tasa.

- newUpDesc: Añade a un array para la actualización de los descuentos, el nuevo porcentaje del descuento, indicando el identificador del descuento en el indice del array.

- updateDiscount: Realiza una llamada *HTTP* al API REST a la ruta */api/createDescuento,* pasando como parámetro el array utilizado en la funciones *newUpDesc*, para que se modifique el porcentaje de los descuentos.

- addDeleteDiscount: Indica en el registro que pertenece al descuento que se le pasa como parámetro en un array de los descuentos de una tasa, que a de ser borrado.

- deleteDiscount: Realiza una llamada *HTTP* al API REST a la ruta */api/deleteDescuento,* pasando como parámetro el array utilizado en la funcion *addDeleteDiscount*, para que se eliminen los descuentos indicados.

También está el controlador *HomeController,* que es el encargado de realizar las funciones para la página principal. Las diferentes funciones de este controlador son:

- goToLogin: Redirige a la página de inicio de sesión.

- loadNum: Realiza una llamada *HTTP* al API REST a la ruta */api/numTasas,* para devolver el número de tasas que tiene el sistema.

- loadedAllTasks: Realiza una llamada *HTTP* al API REST a la ruta */api/bestTasks,* para devolver las cuatro tasas más utilizadas del sistema.

- loadTasas: Realiza una llamada *HTTP* al API REST a la ruta */api/allTasas,* pasándole por parámetro “null” para indicar que no se quiere un departamento específico, y un número que indicará la página que se quiere mostrar, siendo cada página de 7 tasas. Es decir, 1 mostrará las tasas de la 1 a la 7, 2 de la 8 a la 14 y así sucesivamente.

Otro controlador es *LoginController,* que es el encargado de realizar las funciones para el inicio de sesión. Las diferentes funciones de este controlador son:

- loginClicked: Realiza una llamada *HTTP* al API REST a la ruta */api/loginUser,* pasándole como parámetros el NIP de la universidad y la contraseña administrativa, para iniciar sesión en el sistema. En caso de estar en el sistema, el token devuelto por el API REST se guarda en una variable local, para pasarla por la cabecera en todas las futuras llamadas y redirige a la pantalla principal. En caso de pertenecer a la universidad pero no estar registrado en este sistema, redirige a la página de registro, indicando automáticamente el NIP, nombre, apellidos y NIF del usuario. En caso contrario, devuelve un mensaje de error.

- signupClicked: Realiza una llamada *HTTP* al API REST a la ruta */api/newUser,* pasándole como parámetros el NIF, el NIP de la universidad, nombre, apellidos, correo electrónico y la contraseña administrativa, para registrarse en el sistema. En caso de estar el usuario registrado en la universidad, el usuario se registrará en el sistema y el token devuelto por el API REST se guarda en una variable local, para pasarla por la cabecera en todas las futuras llamadas y redirige a la pantalla principal. En caso contrario, se devolverá un mensaje de error.

Por último, esta el controlador *TaskController,* que será el encargado de realizar las funciones correspondientes a los pagos de los usuarios. Las diferentes funciones de este controlador son:

- goMyTasks: Redirige a la página donde se muestran los pagos del usuario.

- loadDepartments: Realiza una llamada *HTTP* al API REST a la ruta */api/allDepartments,* pasándole como parámetro “null” para indicar que devuelva todos los departamentos y no uno específico.

- loadTasas: Realiza una llamada *HTTP* al API REST a la ruta */api/allTasas,* pasándole como parámetro null para indicar que no se quiere un departamento específico y “null” para que devuelva todos las tasas.

- loadTasa: Realiza una llamada *HTTP* al API REST a la ruta */api/getTasa,* para que devuelva la tasa que se le especifica al pasar por parámetro su identificador.

- selectTasa: Selecciona un tasa para ser creada.

- loadPayments: Realiza una llamada *HTTP* al API REST a la ruta */api/allPayments,* para que devuelva las formas de pago que tiene el sistema.

- buscaTasa: Realiza una llamada como en *loadTasas*, pero existe la posibililidad de pasarle por parámetro un departamento específico.

- addVariable: Función para añadir un registro a un array para los atributos. Cada registro tiene los siguientes parámetros: nombre del atributo (*atributo),* el tipo que será “variable”, y el valor del atributo (*valor).* En primer lugar, se comprobará que el atributo no esté ya en el array. Si está el valor se modificará y si no está se añadirá un registro nuevo.

- addSelection: Función para añadir un registro a un array para los atributos. Cada registro tiene los siguientes parámetros: nombre del atributo (*atributo),* el tipo que será “seleccion”, y el valor del atributo (*valor).* En primer lugar, se comprobará que el atributo no esté ya en el array. Si está el valor se modificará y si no está se añadirá un registro nuevo.

- createTaskClicked: Realiza una llamada *HTTP* al API REST a la ruta */api/createTask* pasándole como parámetro el identificador de una tasa y el array utilizado en *addVariable* y *addSelection,* para crear el pago de la tasa y redirigir a la página de pago.

- loadNewTask: Realiza una llamada *HTTP* al API REST a la ruta */api/task* para devolver el pago que se le especifica al pasar por parámetro su identificador.

- loadedMyTasks: Realiza una llamada *HTTP* al API REST a la ruta */api/history* indicando “null” o un número del 0 al 3, para que muestre, o todos los pagos de un usuario, o los pagos de un usuario con el estado al que pertence el número que se le pasa (mirar el atributo *state* de *pipuz_task* en el Modelo)

- payClicked: Redirige a la página del pago que se le pasa por parámetro.

-getCartaPago: Realiza una llamada *HTTP* al API REST a la ruta */api/GetCarta* para devolver el pdf de la carta de pago para el pago que se le especifica al pasar por parámetro su identificador.

- payPaypal: Realiza una llamada *HTTP* al API REST a la ruta */api/payPaypal* para redirigir al link en Paypal del pago que se le especifica al pasar por parámetro su identificador.

- paypalGood: Realiza una llamada *HTTP* al API REST a la ruta */api/paypalGood* para verificar que el pago que se ha realizado en Paypal ha sido correcto y actualizar la base de datos poniendo el pago como pagado.

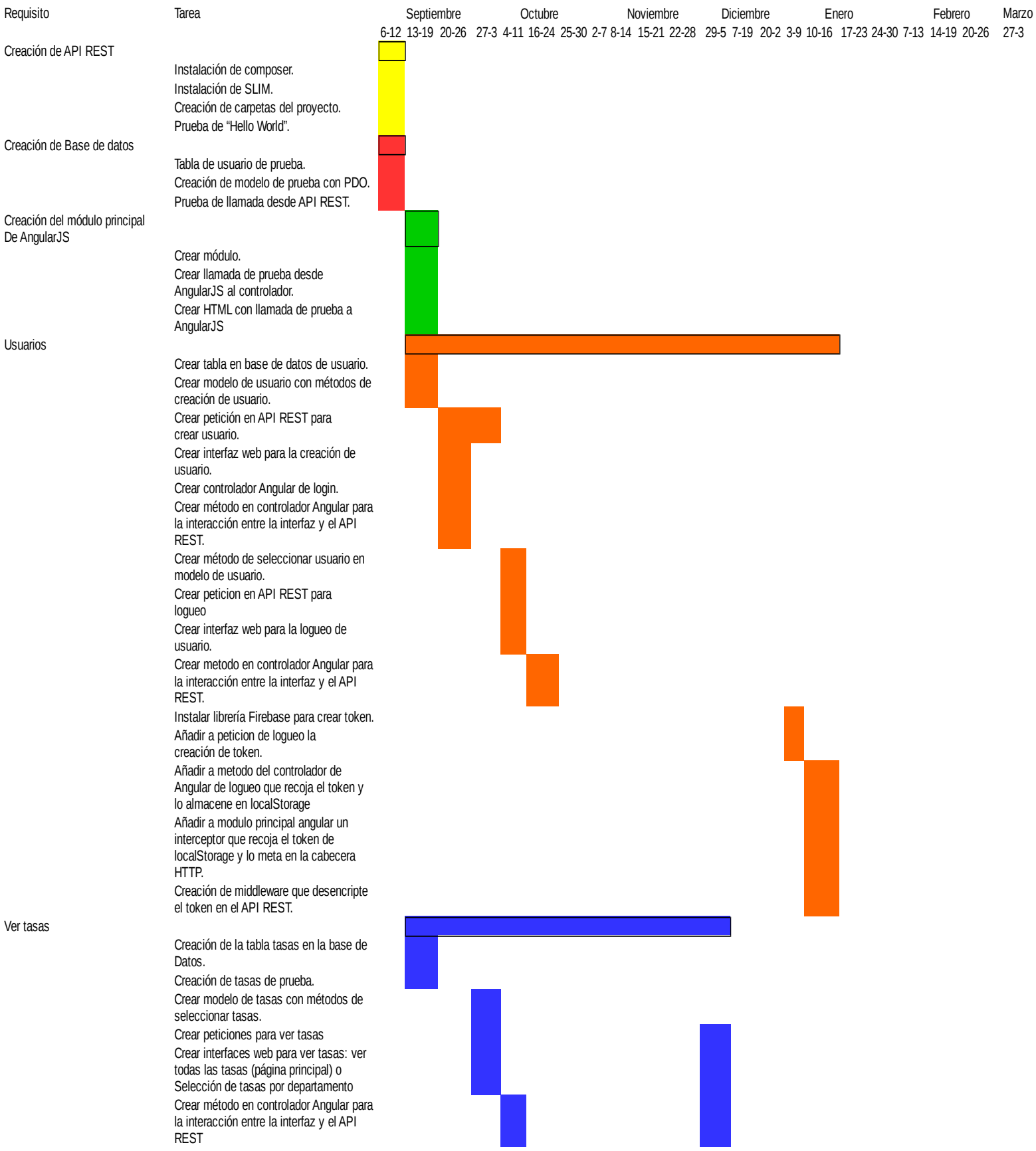
- paypalBad: Realiza una llamada *HTTP* al API REST a la ruta */api/paypalBad* que devuelve un mensaje indicando que ha fallado el pago en Paypal.

- loadTPV: Realiza una llamada *HTTP* al API REST a la ruta */api/tpvData* para devolver un formulario de tipo *hidden*, que tendrá toda la configuración necesaria para que el navegador se redirija al TPV de Ceca.

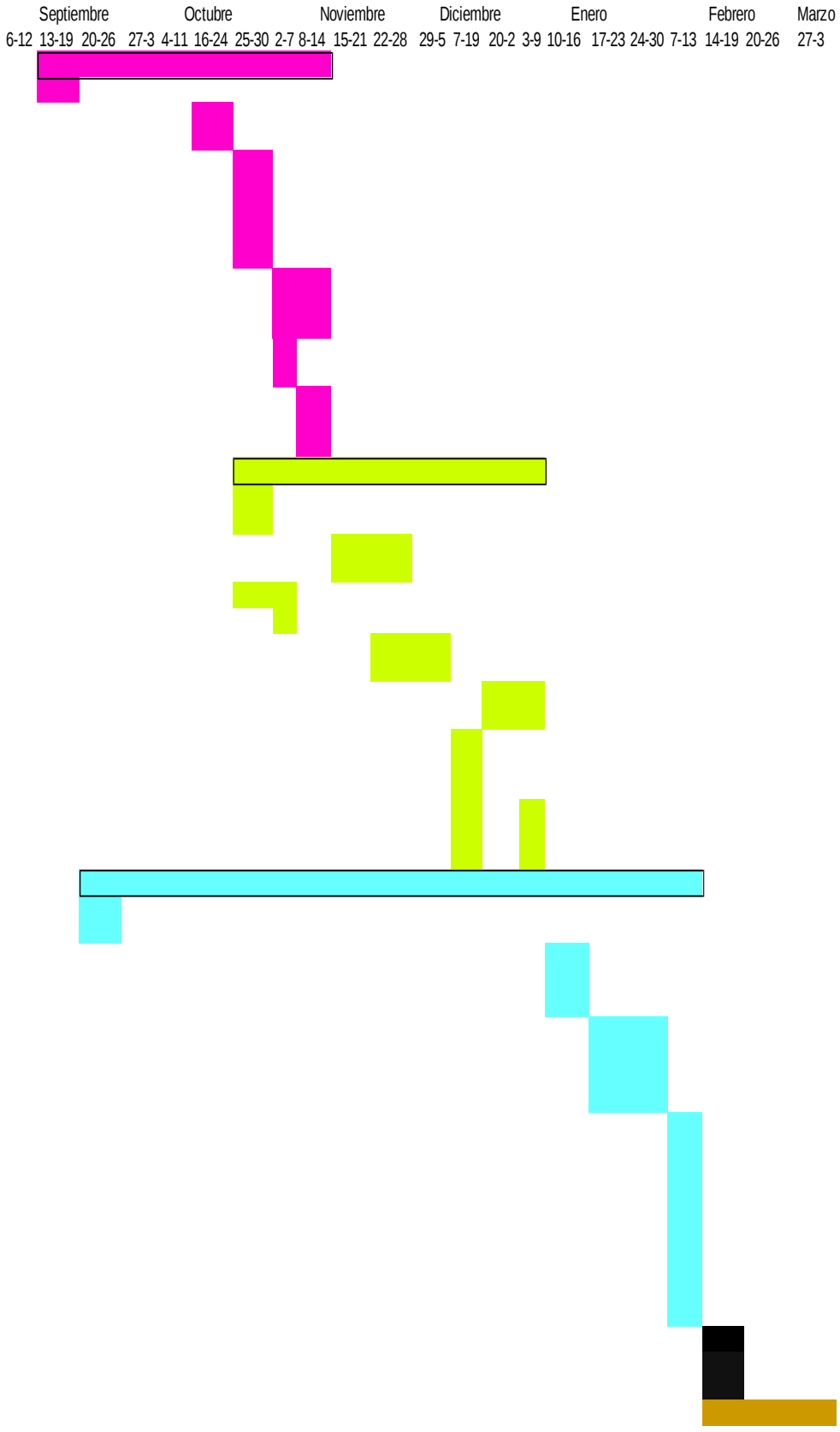
- tpvGood: Realiza una llamada *HTTP* al API REST a la ruta */api/isPay* para verificar que el pago que se ha realizado en TPV ha sido correcto y actualizar la base de datos poniendo el pago como pagado.

- cancelTask: Realiza una llamada *HTTP* al API REST a la ruta */api/cancelTask* para actualizar el estado a cancelado del pago que se le especifica al pasar por parámetro su identificador.

Anexo 8 - Diagrama Gantt



Requisito	Tarea
Crear pagos	
	Creación de tabla task en base de datos.
	Crear modelo de task con método para controlar pagos.
	Crear petición en API REST para crear los pagos que sólo la podrán ejecutar los usuarios registrados.
	Crear petición en API REST para mostrar los pagos del usuario.
	Crear método en controlador Angular para la interacción entre la interfaz y el API REST
	Modificar las interfaces que mostraban tasas para poder crear pagos.
	Crear interfaces web para mostrar los pagos de un usuario (todos los pagos o uno en concreto)
Pagar tasas	
	Configurar consola del TPV de CECA
	Instalar librería del TPV de CECA
	Crear peticiones en API REST para gestionar el TPV
	Crear cuenta y configuración en Paypal
	Instalar librería de Paypal
	Crear peticiones en API REST para gestionar los pagos con Paypal
	Crear petición en API REST para la descarga de pdf de carta de pago
	Modificar interfaz de pago para aplicar Los pagos con TPV, Paypal y Carta de pagos.
	Crear método en controlador Angular para la interacción entre la interfaz y el API REST
Administración	
	Crear tabla en base de datos de administración.
	Crear modelo de administración.
	Crear petición en API REST para mostrar todos los pagos de un departamento
	Crear petición en API REST para la creación de tasas
	Crear petición en API REST para la actualización de tasas.
	Crear petición en API REST para la creación de usuarios.
	Crear petición en API REST para la creación de pagos a usuarios.
	Crear petición en API REST para la actualización del estado de los pagos.
	Crear peticiones en API REST para la creación, actualización y borrado de descuentos.
Seguridad	
	Configurar servidor para utilizar protocolo HTTPS con TLS.
Pruebas	



Anexo 9 - Pruebas

Pruebas del API REST

Las pruebas del *API REST* se han realizado con la herramienta *Postman,* indicando los datos a probar en un archivo *CSV.*

Los datos del archivo *CSV* para cada llamada del *API REST* han sido los siguientes:

/api/loginUser

userid	**password**	**loginCode**
683537	*Mi contraseña administrativa*	2
683537	*Mi contraseña administrativa*	2
683537	*Mi contraseña administrativa*	2
151397	*Mi contraseña administrativa*	2
683537	Aasasa (Contraseña falsa)	1
616586 (usuario que no está en el sistema)	*Su contraseña administrativa*	0

Llamada api/loginUser, donde se inserta el id del usuario y la contraseña administrativa. En los cuatro primeros casos el usuario y contraseña son correctos, con lo que la respuesta será 2. En el quinto caso la contraseña es falsa, por lo que la respuesta será 1. Por último, como el usuario y la contraseña son verdaderos pero no existe dicho usuario en el sistema de *PIPUZ,* la respuesta será 0.

/api/createTask

tasaCreateTask	**createTaskCode**
2	0
90	1
asda	1
	1
-	-
-	-

Llamada api/createTask, donde se inserta el id de la tasa. En el primer caso, al existir la tasa 2, el valor devuelto es 0. En el segundo y tercero, al no existir, devolverá 1. En el cuarto, al ser vacío, devolverá 1 también.

/api/getUser

getUserCode
1
1
1
1
0
0

Llamada api/getUser, para determinar si existe el usuario que ha iniciado sesión en la llamada api/loginUser. Por ello el valor devuelto en los cuatro primeros es 1 y en los dos últimos es 0.

/api/allDepartments

allDepartmentDepart	**allDepartmentCode**	**numDepartments**
null	1	2
5	1	1
89	0	0
Asd4	0	0
-	-	-
-	-	-

Llamada api/allDepartments, indicando el identificador del departamento. Si se escribe *null,* devuelve todo los departamentos (actualmente hay dos) y el código será 1. Si se escribe 5 (hay un departamente con identificador 5) devolverá el código 1 y el número de departamentos será 1. En los demás casos, no existen dichos identificadores con lo que devolverá el código 0 y de número de departamentos 0.

/api/allTasas

allTasaDepart	**allTasasNum**	**numTasas**
5	*null*	7
null	1	7
69	2	0
As	*Asd*	0
-	-	-
-	-	-

Llamada api/allTasas, indicando el identificador del departamento de las tasas y el número de paginación de las tasas que se quieren mostrar (cada página de tasas contiene 7 tasas). En el

primer caso, se selecciona el departamento 5 con número *null,* y por lo tanto cogerá las 7 tasas que hay en el departamento 5 (en este caso sólo hay 7 tasas). En el segundo caso no se selecciona ningún departamento al escribir *null* y se escoge la página 1, con lo cuál el número de tasas será 7. En los demás casos el número de departamentos no existe con lo cúal el número de tasas será 0.

/api/getTasa

getTasaTasa	**getTasaCode**
82	1
5	0
Sdas	1
	1
-	-
-	-

En la llamada api/getTasa, sólo recibirá respuesta positiva (código 0) en el segundo caso (tasa 5) ya que las demás tasas no existen.

/api/task

taskid	**taskCode**
181	0
18692	1
Asd	1
	1
-	-
-	-

En la llamada api/task, sólo recibirá respuesta positiva (código 0) en el primer caso (pago 181) ya que las demás pagos no existen.

/api/history

history	**historyCode**
0	0
6	1
Asd	1
	1
-	-
-	-

En la llamada api/history, sólo recibirá respuesta positiva (código 0) en el primer caso (valor de option 0) ya que los demás valores no son posibles (únicamente 0, 1, 2 y 3)

/api/cancelTask

cancelTaskCode
0
1
1
1
-
-

En la llamada api/cancelTask, sólo recibirá respuesta positiva (código 0) en el primer caso ya que en la llamada anterior api/createTask, únicamente se ha creado el primer caso.

/api/getAdminDepartments

numAdminDepartments
1
1
1
-
-
-

En la llamada api/getAdminDepartments, sólo recibirán respuesta los tres primeros casos, ya que los usuarios son administradores.

/api/getAdminTasks

departmentAdminTasks	**orderAdminTasks**	**filtroAdminTasks**	**numAdminTasks**
5	*name asc*	*null*	1
6	*null*	*null*	0
5	*asdf*	*asdfl*	0
-	-	-	-
-	-	-	-
-	-	-	-

En la llamada api/getAdminTask, sólo recibirá respuesta positiva (código 1) en el primer caso ya que se escriben correctamente todos los parámetros.

/api/createTasa

departmentCreateTasa	**nameCreateTasa**	**descriptionCreateTasa**	**prizeCreateTasa**
5	*Curso de fútbol*	*Curso de fútbol*	50
5	*Curso de baloncesto*	*Curso de baloncesto*	
asd			52asd
-	-	-	-
-	-	-	-
-	-	-	-
expireCreateTasa	**valor1CreateTasa**	**opt1CreateTasa**	**value1CreateTasa**
7	1	*a_academico*	*2017/2018*
7	1		
siete	2	*asd*	
-	-	-	-
-	-	-	-
-	-	-	-
valor2CreateTasa	**opt2CreateTasa**	**sel21CreateTasa**	**sel22CreateTasa**
2	*a_contable*	2017	2018
	a_contable		2018
	123		
-	-	-	-
-	-	-	-
-	-	-	-
valor3CreateTasa	**opt3CreateTasa**	**codeCreateTasa**	
0	*experimentalidad*	1	
	a_academico	0	
		0	
-	-	-	
-	-	-	
-	-	-	

En la llamada api/createTasa, sólo recibirá respuesta positiva (código 1) en el primer caso ya que se escriben correctamente todos los parámetros.

/api/updateTasa

descriptionUpdateTasa	**prizeUpdateTasa**	**expireUpdateTasa**	**valor1UpdateTasa**
Curso de fútbol	40	7	Único
Curso de baloncesto			1

	asd		
-	-	-	-
-	-	-	-
-	-	-	-
opt1UpdateTasa	**value1UpdateTasa**	**valor2UpdateTasa**	**opt2UpdateTasa**
a_academico	*2017/2018*	*Selección*	*experimentalidad*
56sad			*asda25*
	2017/2018	5	
-	-	-	-
-	-	-	-
-	-	-	-
sel21UpdateTasa	**sel22UpdateTasa**	**valor3UpdateTasa**	**opt3UpdateTasa**
Sí	*No*	*Variable*	*a_contable*
			aasda62
		256	
-	-	-	-
-	-	-	-
-	-	-	-
optUp1	**valorUp1**	**valueUp1**	**optUp2**
convocatoria	1	*Única*	*repetición*
asd55a41		23	
	23		*asd45a*
-	-	-	-
-	-	-	-
-	-	-	-
valorUp2	**selUp21**	**selUp22**	**codeUpdateTasa**
2	*Sí*	*No*	0
			1
25			1
-	-	-	-
-	-	-	-
-	-	-	-

En la llamada api/updateTasa, sólo recibirá respuesta positiva (código 1) en el primer caso ya que se escriben correctamente todos los parámetros.

/api/createDescuento

createDescuentoName1	**createDescuentoPorcentaje1**	**createDescuentoName2**
Familia numerosa	25	*Discapacidad*

	29	*Curso de baloncesto*
	2018	
-	-	-
-	-	-
-	-	-
createDescuentoPorcentaje2	**createDescuentoDepartment**	**createDescuentoTasa**
50	7	4
Discapacidad	20	389
	asd	asd
-	-	-
-	-	-
-	-	-

createDescuentoCode
0
1
1
-
-
-

En la llamada api/createDescuento, sólo recibirá respuesta positiva (código 1) en el primer caso ya que se escriben correctamente todos los parámetros.

/api/updateDescuento

updateDescuentoPorcentaje1	**updateDescuentoPorcentaje2**	**updateDescuentoCode**
30	60	0
25	35	1
asd		1
-	-	-
-	-	-
-	-	-

En la llamada api/updateDescuento, sólo recibirá respuesta positiva (código 1) en el primer caso ya que se escriben correctamente todos los parámetros y en el segundo caso, aunque estén bien escritos, no existe el descuento, ya que no se ha creado en la llamada anterior.

/api/deleteDescuento

deleteDescuentoCode
0

1
1
-
-
-

En la llamada api/deleteDescuento, sólo recibirá respuesta positiva (código 0) en el primer caso ya que en los demás no se ha creado el descuento en la llamada api/createDescuento.

Pruebas de la interfaz

Creación de usuario	**Resultado esperado**	**Resultado obtenido**
Crear usuario existente.	Mensaje de que usuario ya está creado en el sistema.	Mensaje de que usuario ya está creado en el sistema.
Crear usuario existente en la universidad con datos consistentes de la universidad.	Usuario creado con sesión iniciada.	Usuario creado con sesión iniciada.
Crear usuario existente en la universidad con datos diferentes a los de la universidad.	Mensaje indicando que los datos son erróneos.	Mensaje indicando que los datos son erróneos.
Crear usuario no existente en la universidad.	Mensaje indicando que el usuario no existe en la universidad con enlace para crearlo.	Mensaje indicando que el usuario no existe en la universidad con enlace para crearlo.

Inicio de sesión	**Resultado esperado**	**Resultado obtenido**
Usuario existente en el sistema y contraseña administrativa de la universidad.	Inicio de sesión correcto y se redirige a la pantalla principal.	Inicio de sesión correcto y se redirige a la pantalla principal.
Usuario existente en el sistema y contraseña administrativa de la universidad errónea.	Mensaje de contraseña errónea.	Mensaje de contraseña errónea.
Usuario no existente en el sistema pero sí en la universidad y con contraseña administrativa.	Redirección a la página de creación de usuario, con los datos de la universidad escritos en los campos correspondientes.	Redirección a la página de creación de usuario, con los datos de la universidad escritos en los campos correspondientes.

Usuario no existente en el sistema pero sí en la universidad y con contraseña administrativa errónea.	Mensaje de contraseña errónea.	Mensaje de contraseña errónea.
Usuario no existente en el sistema y en la universidad.	Mensaje indicando que el usuario no existe en la universidad con enlace para crearlo.	Mensaje indicando que el usuario no existe en la universidad con enlace para crearlo.

Buscar tasa	**Resultado esperado**	**Resultado obtenido**
Escribir texto que no coincide con ninguna tasa.	No se muestra ningún resultado.	No se muestra ningún resultado.
Escribir texto que coincide con alguna tasa.	Se muestran los resultados que coinciden.	Se muestran los resultados que coinciden.

Seleccionar una tasa	**Resultado esperado**	**Resultado obtenido**
El usuario ha iniciado sesión.	Redirección a la página de crear pago de la tasa seleccionada.	Redirección a la página de crear pago de la tasa seleccionada.
El usuario no ha iniciado sesión.	Redirección a la página de inicio de sesión.	Redirección a la página de inicio de sesión.

Crear un pago	**Resultado esperado**	**Resultado obtenido**
Se insertan todos los atributos de la tasa.	Pago creado y redirección a la página para pagar o cancelar el pago.	Pago creado y redirección a la página para pagar o cancelar el pago.
Falta algún atributo de la tasa.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.

Ver un pago	**Resultado esperado**	**Resultado obtenido**
Ver un pago por realizar.	Redirección a la página para pagar o cancelar el pago.	Redirección a la página para pagar o cancelar el pago.
Ver un pago realizado.	Redirección a la página que muestra el pago y su estado.	Redirección a la página que muestra el pago y su estado.
Ver un pago caducado.	Redirección a la página que muestra el pago y su estado.	Redirección a la página que muestra el pago y su estado.
Ver un pago cancelado.	Redirección a la página que muestra el pago y su estado.	Redirección a la página que muestra el pago y su estado.

Pagar un pago	Resultado esperado	Resultado obtenido
Elegir forma de pago TPV.	Redirección a la consola del TPV.	Redirección a la consola del TPV.
Elegir forma de pago PayPal.	Redirección a la página de PayPal.	Redirección a la página de PayPal.
Elegir forma de pago Carta de Pago.	Descarga de documento PDF con la carta de pago.	Descarga de documento PDF con la carta de pago.

Pago en TPV	Resultado esperado	Resultado obtenido
Escribir correctamente datos de la tarjeta.	Pago realizado con éxito.	Pago realizado con éxito.
Escribir incorrectamente datos de la tarjeta.	Pago no realizado.	Pago no realizado.

Pago en PayPal	Resultado esperado	Resultado obtenido
Iniciar sesión en PayPal correctamente y pulsar en continuar.	Pago realizado con éxito.	Pago realizado con éxito.
Iniciar sesión en PayPal incorrectamente.	Pago no realizado.	Pago no realizado.

Cancelar un pago	Resultado esperado	Resultado obtenido
Dar botón cancelar pago.	Pago cancelado.	Pago cancelado.

Cerrar sesión	Resultado esperado	Resultado obtenido
Dar botón de cerrar sesión.	El usuario se queda sin la sesión iniciada y redirección a la pantalla principal.	El usuario se queda sin la sesión iniciada y redirección a la pantalla principal.

Acceder a módulo de administración	Resultado esperado	Resultado obtenido
Dar botón de Administración (sólo aparece en caso de que seas administrador).	Acceso al módulo de administración.	Acceso al módulo de administración.
Acceder a la URL https://nuez.unizar.es/~pipuz/admin siendo administrador.	Acceso al módulo de administración.	Acceso al módulo de administración.
Acceder a la URL	Cierre de sesión y redirección	Cierre de sesión y redirección

https://nuez.unizar.es/~pipuz/admin sin ser administrador.	a página de inicio de sesión.	a página de inicio de sesión.

Buscar pago desde administración	**Resultado esperado**	**Resultado obtenido**
Escribir texto que no coincide con ningún nombre, descripción o usuario.	No se muestra ningún resultado.	No se muestra ningún resultado.
Escribir texto que coincide con algún nombre, descripción o usuario.	Se muestran los resultados que coinciden.	Se muestran los resultados que coinciden.

Crear tasa	**Resultado esperado**	**Resultado obtenido**
Crear tasa escribiendo todos los campos.	Tasa creada correctamente.	Tasa creada correctamente.
Crear tasa dejando algún campo en blanco.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Crear tasa añadiendo un atributo sin nombre del atributo.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Crear una tasa añadiendo un atributo único sin valor del atributo.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Crear una tasa añadiendo un atributo de selección sin posibles valor del atributo.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Crear una tasa añadiendo un atributo de selección con un posible valor del atributo en blanco.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Crear una tasa añadiendo dos atributos que tengan el mismo nombre.*	No se tiene que volver a mostrar el mismo nombre.	Se crea el último atributo que se ha creado.
Crear una tasa con todos los campos y con los atributos bien formados.	Tasa creada correctamente.	Tasa creada correctamente.

* Se ha corregido el error

Actualizar tasa	Resultado esperado	Resultado obtenido
Actualizar tasa dejando escritos todos los campos.	Tasa actualizada correctamente.	Tasa actualizada correctamente.
Actualizar tasa dejando algún campo en blanco.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Actualizar tasa dejando un atributo sin nombre del atributo.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Actualizar una tasa dejando un atributo único sin valor del atributo.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Actualizar una tasa dejando un atributo de selección sin posibles valor del atributo.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Actualizar una tasa dejando un atributo de selección con un posible valor del atributo en blanco.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Actualizar una tasa dejando dos atributos que tengan el mismo nombre.*	No se tiene que volver a mostrar el mismo nombre.	Se crea el último atributo que se ha creado.
Actualizar una tasa con todos los campos y con los atributos bien formados.	Tasa actualizada correctamente.	Tasa actualizada correctamente.

* Se ha corregido el error.

Crear un usuario	Resultado esperado	Resultado obtenido
Escribir todos los campos correctamente y coincidiendo con los datos de la universidad.	Usuario creado correctamente y envío de correo al email indicado.	Usuario creado correctamente y envío de correo al email indicado.
Escribir todos los campos correctamente y que no coincidan con los datos de la universidad.	Mensaje de que algún campo no es correcto.	Mensaje de que algún campo no es correcto.
Dejarse algún campo por rellenar.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.

Generar recibo	Resultado esperado	Resultado obtenido
Escribir usuario que esta registrado en el sistema.	Pago creado correctamente, con envío de correo al email del usuario.	Pago creado correctamente, con envío de correo al email del usuario.
Escribir usuario que no esta registrado en el sistema.	No deja crear el pago.	No deja crear el pago.
Se insertan todos los atributos de la tasa.	Pago creado correctamente, con envío de correo al email del usuario.	Pago creado correctamente, con envío de correo al email del usuario.
Falta algún atributo de la tasa.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.

No escribir usuario.	No deja crear el pago.	No deja crear el pago.

Crear pago al contado	**Resultado esperado**	**Resultado obtenido**
Escribir usuario que esta registrado en el sistema.	Pago creado correctamente, con envío de correo al email del usuario.	Pago creado correctamente, con envío de correo al email del usuario.
Escribir usuario que no esta registrado en el sistema.	No deja crear el pago.	No deja crear el pago.
Se insertan todos los atributos de la tasa.	Pago creado correctamente, con envío de correo al email del usuario.	Pago creado correctamente, con envío de correo al email del usuario.
Falta algún atributo de la tasa.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
No escribir usuario.	No deja crear el pago.	No deja crear el pago.

Actualizar Carta de pago	**Resultado esperado**	**Resultado obtenido**
Escribir usuario que esta registrado en el sistema.	Pago actualizado.	Pago actualizado.
Escribir usuario que no esta registrado en el sistema.	Mensaje de error.	Mensaje de error.
No escribir usuario.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Escribir una fecha de pago en formato dd-mm-yyyy	Pago actualizado.	Pago actualizado.
Escribir una fecha de pago sin formato dd-mm-yyyy	Mensaje de error.	Mensaje de error.
No escribir fecha de pago.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Escribir un identificador del pago correcto.	Pago actualizado.	Pago actualizado.
Escribir un identificador del pago que no existe.	Mensaje de error.	Mensaje de error.
Escribir un identificador del pago que no esté en estado por pagar.	Mensaje de error.	Mensaje de error.
No escribir identificador del	Mensaje de que faltan campos	Mensaje de que faltan campos

pago.	por rellenar.	por rellenar.

Crear descuento	**Resultado esperado**	**Resultado obtenido**
Crear un descuento con nombre y porcentaje correcto.	Descuento creado correctamente.	Descuento creado correctamente.
Crear un descuento sin nombre.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Crear un descuento sin porcentaje.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Crear un descuento con porcentaje incorrecto (número negativo o mayor que 100 o letras).	Mensaje de error.	Mensaje de error.

Actualizar descuento	**Resultado esperado**	**Resultado obtenido**
Dejar un porcentaje en blanco.	Mensaje de que faltan campos por rellenar.	Mensaje de que faltan campos por rellenar.
Dejar un porcentaje incorrecto (número negativo o mayor que 100 o letras).	Mensaje de error.	Mensaje de error.
Dejar correctamente los porcentajes.	Porcentaje actualizado correctamente.	Porcentaje actualizado correctamente.

Borrar un descuento	**Resultado esperado**	**Resultado obtenido**
Dar botón rojo de borrar y pulsar sobre *Borrar*.	Descuento borrado.	Descuento borrado.