




**Universidad
Zaragoza**

Historia Clínica Especificación de prototipos

Pablo Doñate Navarro
Adnana Dragut
Proyecto Software

Versión: 1.0
Fecha: 03/05/2022

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	---	---

HOJA DE CONTROL

Proyecto	Historia Clínica		
Entregable	Diseño		
Autor	Pablo Doñate Navarro y Adnana Dragut		
Versión/Edición	1.0	Fecha Versión	03/05/2022
Aprobado por	Sin aprobación	Fecha Aprobación	-
		Nº Total de Páginas	19

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
1.0	Versión inicial.	Pablo Doñate Navarro Adnana Dragut	03/05/2022

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos
Pablo Doñate Navarro
Adnana Dragut



 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	---	---

TABLA DE CONTENIDOS

<i>Especificación de prototipos</i>	4
<i>Prototipos App cliente</i>	4
<i>Prototipos App Servidor</i>	12

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

1. Especificación de prototipos

En este documento se detallarán el trabajo que se ha llevado a cabo en cada uno de los prototipos definidos en la planificación. Para ello, en primer lugar se comentarán los prototipos de la aplicación cliente y posteriormente se comentarán los prototipos de la aplicación servidor.

1.1. *Prototipos App cliente*

Los prototipos definidos en la planificación son los siguientes:

Prototipo 1: Funcionalidad Login


En este prototipo se ha implementado toda la funcionalidad necesaria para que un usuario pueda acceder a la aplicación. Para poder realizar el inicio de sesión el usuario deberá introducir su dni, su correo electrónico y su contraseña. Toda la funcionalidad se ha implementado en el sub-prototipo 1.1.

Prototipo 1.1: Crear vista Login y permitir obtener un sanitario

En este sub-prototipo se ha llevado a cabo la implementación de la vista necesaria para que un usuario pueda introducir sus credenciales y realizar el inicio de sesión, así como todos los métodos necesarios para realizar las verificaciones oportunas, para leer la entrada del usuario, y para mandar/recibir peticiones por parte del servidor. Así mismo, aunque en la planificación no queda reflejado, se ha llevado a cabo la implementación de una vista adicional utilizada para el arranque de la aplicación.

Las tareas más importantes que se han llevado a cabo en el prototipo 1.1 son las siguientes:

- Creación de la clase “WelcomeVista.java”, utilizando para ello el GUI Builder de Java Swing, así como todos los métodos utilizados en la clase para simular la conexión de la aplicación cliente con el servidor.
- Creación de la clase “LoginVista.java”, utilizando para ello el GUI Builder de Java Swing, así como los métodos de la clase que permiten realizar acciones importantes como mostrar el estado de la conexión con el servidor, capturar la entrada del usuario, o enviar la solicitud de inicio de sesión a la capa control.
- Creación de la clase “Comms.java” con los métodos necesarios para

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

establecer la conexión con el servidor, para desconectarse del servidor, y para recibir respuestas del servidor en cuanto a la solicitud de conexión.

- Creación de la clase “UsuarioDTO” con sus atributos, métodos getter-setter, así como los métodos toJson() y toString().
- Creación de la clase “ProxyUsuario” con el método para enviar la solicitud de verificación de un usuario al servidor.
- Creación de las clases restantes necesarias para poder intercambiar información entre las tres capas de MVC, que son: “OyenteServidor.java”, “ConexionPushHospital.java”, “PrimitivaComunicacion.java”, “Config.java”, “Hospital.java”, “OyenteVista.java”. Cabe destacar que en este prototipo únicamente se han implementado los métodos necesarios para establecer la conexión con el servidor y para poder verificar a un usuario que inició sesión, el resto de las funcionalidades se han incorporado de manera gradual en sus prototipos correspondientes.
- Implementación del resto de funcionalidad adicional como volver atrás o poder hacer visible la contraseña introducida.

Prototipo 2: Funcionalidad Menú Admin


En este prototipo se ha implementado toda la funcionalidad correspondiente al usuario administrador. Concretamente se han definido las clases necesarias para que un usuario administrador pueda dar de alta nuevos sanitarios, pueda eliminar sanitarios existentes y pueda editar la información de dichos usuarios.

Prototipo 2.1: Crear vistas de Menú Admin, Nuevo Sanitario y Editar Sanitario

En este sub-prototipo se ha llevado a cabo la implementación de todas las vistas necesarias para el prototipo 2, sin incluir ningún tipo de funcionalidad.

Las tareas más importantes que se han llevado a cabo en el prototipo 2.1 son las siguientes:

- Creación de la clase “MenuAdminVista.java” utilizando para ello el GUI Builder de Java Swing.
- Creación de la clase “NuevoSanitarioVista.java” utilizando para ello el GUI Builder de Java Swing.

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

- Creación de la clase “EditarSanitarioVista.java” utilizando para ello el GUI Builder de Java Swing.

Prototipo 2.2: Permitir añadir nuevo sanitario

En este sub-prototipo se ha llevado a cabo la implementación de todas las clases y métodos necesarios en las tres capas, para poder añadir un nuevo sanitario.

Las tareas más importantes que se han llevado a cabo en el prototipo 2.2 son las siguientes:


- Implementación de los métodos de la clase “NuevoSanitarioVista.java” que permiten realizar acciones importantes como mostrar el estado de la conexión con el servidor, capturar la entrada del usuario, enviar la solicitud para dar de alta un sanitario a la capa control o recibir respuesta por parte de la capa modelo tras enviar dicha solicitud.
- Creación de la clase “SanitarioDTO” con sus atributos, métodos getter-setter, así como los métodos toJson() y toString().
- Creación de la clase “ProxySanitario” con el método para enviar la solicitud de dar de alta un sanitario al servidor.
- Implementación en la clase “Comms.java” del método para recibir la respuesta por parte del servidor tras solicitar dar de alta un sanitario.
- Implementación del resto de funcionalidad adicional como volver atrás o verificar la información introducida.

Prototipo 2.3: Permitir editar un sanitario

En este sub-prototipo se ha llevado a cabo la implementación de todas las clases y métodos necesarios en las tres capas, para poder eliminar un sanitario y editar su información.

Las tareas más importantes que se han llevado a cabo en el prototipo 2.3 son las siguientes:

- Implementación de los métodos de la clase “EditarSanitarioVista.java” que permiten realizar acciones importantes como mostrar el estado de la conexión con el servidor, obtener sanitarios existentes, enviar la solicitud para eliminar un sanitario del sistema a la capa control,

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

capturar los campos modificados por el usuario, enviar la solicitud para modificar un sanitario, recibir respuesta por parte de la capa modelo tras solicitar eliminar un sanitario o recibir respuesta por parte de la capa modelo tras solicitar modificar un sanitario.

- Creación de los métodos en la clase “ProxySanitario.java” que permiten obtener la lista con sanitarios, eliminar un sanitario y modificar un sanitario.
- Creación de los métodos en la clase “Comms.java” para recibir la respuesta por parte del servidor tras solicitar eliminar un sanitario y modificar un sanitario.
- Implementación del resto de funcionalidad adicional como volver atrás u obtener el sanitario seleccionado.

Prototipo 3: Funcionalidad Menú Paciente

En este prototipo se ha implementado toda la funcionalidad correspondiente al usuario sanitario. Concretamente se han definido las clases necesarias para que un usuario administrador pueda dar de alta nuevos pacientes, para añadir un nuevo episodio a un paciente, para añadir un diagnóstico al episodio de un paciente, para crear/eliminar la cita de un paciente, para ver/añadir vacunas a un paciente, para ver/añadir medicamentos a la receta electrónica de un paciente y para ver la historia con la información completa de un paciente.

Prototipo 3.1: Crear vistas de Menú Paciente y Nuevo Paciente


En este sub-prototipo se ha llevado a cabo la implementación de las vistas de los dos menús principales del prototipo 3, sin incluir ningún tipo de funcionalidad.

Las tareas más importantes que se han llevado a cabo en el prototipo 3.1 son las siguientes:

- Creación de la clase “MenuSanitarioVista.java” utilizando para ello el GUI Builder de Java Swing.
- Creación de la clase “NuevoPacienteVista.java” utilizando para ello el GUI Builder de Java Swing.

Prototipo 3.2: Permitir añadir nuevo paciente

En este sub-prototipo se ha llevado a cabo la implementación de todas las

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

clases y métodos necesarios en las tres capas, para poder añadir un nuevo paciente.

Las tareas más importantes que se han llevado a cabo en el prototipo 3.2 son las siguientes:


- Implementación de los métodos de la clase “NuevoPacienteVista.java” que permiten realizar acciones importantes como mostrar el estado de la conexión con el servidor, capturar los campos introducidos por el usuario, enviar la solicitud para añadir un nuevo paciente al sistema a la capa control, y recibir respuesta por parte de la capa modelo tras solicitar añadir un nuevo paciente.
- Creación de la clase “PacienteDTO” con sus atributos, métodos getter-setter, así como los métodos toJson() y toString().
- Creación de la clase “ProxyPaciente” con el método para enviar la solicitud para añadir un nuevo paciente al sistema.
- Creación de los métodos en la clase “Comms.java” para recibir la respuesta por parte del servidor tras solicitar añadir un nuevo paciente al sistema.
- Implementación del resto de funcionalidad adicional como volver atrás o verificar la información introducida.

Prototipo 3.3: Crear vista Gestión de Paciente y permitir obtener lista de pacientes

En este sub-prototipo se ha llevado a cabo la implementación de la vista necesaria para poder realizar las operaciones con un paciente, así como del método necesario para obtener la lista de pacientes.

Las tareas más importantes que se han llevado a cabo en el prototipo 3.3 son las siguientes:

- Creación de la clase “MenuGestionPacientesVista.java” utilizando para ello el GUI Builder de Java Swing.
- Creación del método para obtener los pacientes del sistema en la clase “ProxyPaciente”.
- Implementación de los métodos de la clase “MenuGestionPacientesVista.java” que permiten realizar acciones importantes como mostrar el estado de la conexión con el servidor y

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

mostrar la lista de pacientes del sistema.

- Implementación del resto de funcionalidad adicional como volver atrás.

Prototipo 3.4: Crear vista Nuevo Episodio y permitir añadir un nuevo episodio.

En este sub-prototipo se ha llevado a cabo la implementación de la vista necesaria para poder crear un nuevo episodio, así como del método necesario para añadir el episodio al paciente.


Las tareas más importantes que se han llevado a cabo en el prototipo 3.4 son las siguientes:

- Creación de la clase “NuevoEpisodioVista.java” utilizando para ello el GUI Builder de Java Swing.
- Implementación de los métodos de la clase “NuevoEpisodioVista.java” que permiten realizar acciones importantes como mostrar el estado de la conexión con el servidor, capturar los campos introducidos por el usuario, enviar la solicitud para añadir un nuevo episodio a la capa control, y recibir respuesta por parte de la capa modelo tras solicitar añadir un nuevo episodio.
- Creación de la clase “EpisodioDeAtencionDTO” con sus atributos, métodos getter-setter, así como los métodos toJson() y toString().
- Creación de la clase “ProxyEpisodio” con el método para enviar la solicitud para añadir un nuevo episodio a un paciente del sistema.
- Creación de los métodos en la clase “Comms.java” para recibir la respuesta por parte del servidor tras solicitar añadir un nuevo episodio.
- Implementación del resto de funcionalidad adicional como volver atrás o verificar la información introducida.

Prototipo 3.5: Crear vista Episodios Paciente y permitir añadir un diagnóstico

En este sub-prototipo se ha llevado a cabo la implementación de la vista necesaria para poder ver los episodios de un paciente, así como del método necesario para añadir un diagnóstico a un episodio no cerrado.

Las tareas más importantes que se han llevado a cabo en el prototipo 3.5 son

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

las siguientes:


- Creación de la clase “EpisodiosPacienteVista.java” utilizando para ello el GUI Builder de Java Swing.
- Creación del método para obtener los episodios de un paciente en la clase “ProxyPaciente”.
- Implementación de los métodos de la clase “EpisodiosPacienteVista.java” que permiten realizar acciones importantes como mostrar el estado de la conexión con el servidor, obtener la lista de episodios de un paciente, capturar los campos introducidos por el usuario, enviar la solicitud para añadir un diagnóstico a un episodio, y recibir respuesta por parte de la capa modelo tras solicitar añadir un diagnóstico.
- Creación del método para añadir un diagnóstico a un episodio de un paciente en la clase “ProxyEpisodio”.
- Creación de los métodos en la clase “Comms.java” para recibir la respuesta por parte del servidor tras solicitar añadir un diagnóstico a un episodio de un paciente.
- Implementación del resto de funcionalidad adicional como volver atrás u obtener el episodio seleccionado.

Prototipo 3.6: Crear vista Nueva Cita y permitir añadir una nueva cita.

En este sub-prototipo se ha llevado a cabo la implementación de la vista necesaria para poder crear una nueva cita de un paciente, así como del método necesario para añadir una nueva cita a un paciente del sistema.

Las tareas más importantes que se han llevado a cabo en el prototipo 3.6 son las siguientes:

- Creación de la clase “NuevaCitaVista.java” utilizando para ello el GUI Builder de Java Swing.
- Creación de la clase “CitaPacienteDTO” con sus atributos, métodos getter-setter, así como los métodos toJson() y toString().
- Creación de la clase “ProxyCitaPaciente” con el método para enviar la solicitud para añadir una nueva cita a un paciente del sistema.

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---


- Implementación de los métodos de la clase “NuevaCitaVistaa.java” que permiten realizar acciones importantes como mostrar el estado de la conexión con el servidor, capturar los campos introducidos por el usuario, enviar la solicitud para añadir una nueva cita, y recibir respuesta por parte de la capa modelo tras solicitar añadir una nueva cita.
- Creación de los métodos en la clase “Comms.java” para recibir la respuesta por parte del servidor tras solicitar añadir una nueva cita.
- Implementación del resto de funcionalidad adicional como volver atrás y verificar la información introducida.

Prototipo 3.7: Crear vista Citas Paciente y permitir eliminar una cita.

En este sub-prototipo se ha llevado a cabo la implementación de la vista necesaria para poder ver las citas de un paciente, así como del método necesario para eliminar una cita de un paciente.

Las tareas más importantes que se han llevado a cabo en el prototipo 3.7 son las siguientes:


- Creación de la clase “CitasPacienteVista.java” utilizando para ello el GUI Builder de Java Swing.
- Creación del método para obtener las citas de un paciente en la clase “ProxyPaciente”.
- Implementación de los métodos de la clase “CitasPacienteVista.java” que permiten realizar acciones importantes como mostrar el estado de la conexión con el servidor, obtener la lista de citas de un paciente, enviar la solicitud para eliminar una cita de un paciente, y recibir respuesta por parte de la capa modelo tras solicitar eliminar una cita.
- Creación del método para eliminar una cita de un paciente en la clase “ProxyCitaPaciente”.
- Creación de los métodos en la clase “Comms.java” para recibir la respuesta por parte del servidor tras solicitar eliminar una cita de un paciente.
- Implementación del resto de funcionalidad adicional como volver atrás u obtener la cita seleccionada.

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

Prototipo 3.8: Crear vista Registro Vacunación y permitir añadir una nueva vacuna

Prototipo 3.9: Crear vista Receta Electrónica y permitir añadir-eliminar un medicamento

Prototipo 3.10: Crear vista Historia Completa y permitir obtener la info completa de un paciente

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

1.2. *Prototipos App Servidor*

Los prototipos definidos en la planificación son los siguientes:

Prototipo 1: Funcionalidad login de Sanitario y Administrador.


Este prototipo, está formado por 4 sub-prototipos, y hace referencia a la parte de inicio de sesión de un usuario administrador o un usuario sanitario, y en caso de no existir, dar de alta un sanitario o modificarlo. Un administrador no podrá ser dado de alta desde la interfaz. Por defecto, en el arranque del servidor se comprueba si existe el administrador, y si no existe, lo da de alta.

A continuación, se van a indicar los ficheros modificados o creados en cada sub-prototipo para llevar a cabo la tarea indicada.

Prototipo 1.1 y 1.3: Permitir inicio de sesión de una administrador y permitir añadir nuevo sanitario.

En el prototipo 1.1, estaba definido dar de alta un sanitario, donde se va a permitir dar de alta a un nuevo sanitario en el sistema. Para ello, antes ha sido necesaria la implementación del prototipo 1.3 (permitir inicio de sesión de un administrador), por lo que se comentarán a continuación ambos:

- Una primitiva de comunicación VERIFICAR_USUARIO y la primitiva de comunicación DAR ALTA SANITARIO, que permiten reconocer las solicitudes recibidas por parte del cliente.
- Han sido necesarias las clases UsuarioDTO y SanitarioDTO para llevar a cabo estos prototipos.
- En la clase ServidorHospital, se han creado dos métodos.
El primero, verificarUsuario(), que lee del socket el usuario en formato JSON a verificar, y devuelve null si el usuario no está dado de alta, y en caso contrario, devolverá el usuario al servidor, indicando si es administrador, o es un sanitario.
El segundo método, darAltaSanitario(), que lee del socket el sanitario a dar de alta en formato JSON. Finalmente, devuelve si ha sido dado de alta correctamente.


 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

- En la clase servidorSanitarios, como en la clase anterior, se han creado dos métodos.
El primero, verificarUsuario(UsuarioDTO _usuario), solicita a la base de datos que verifique la existencia del usuario.
El segundo, añadirSanitario(SanitarioDTO _sanitario), primero solicita a la DB el alta del sanitario como usuario, y luego ya lo especializa en sanitario (ya que puede existir un usuario administrador). Finalmente, notifica a todos los sanitarios activos el alta del nuevo sanitario.
- En la clase que contiene el acceso a la base de datos, DataBaseControl, ha sido necesaria la implementación de los siguientes métodos.
El primer método, verificarUsuario(UsuarioDTO _usuario), que realiza la ejecución de la consulta Select, y comprueba si dicho usuario existe ya en la base de datos, devolviendo la instancia del mismo.
El segundo método, addUsuario(String _dni, String _email, String _pwd), realiza la ejecución de la consulta Insert into para llevar a cabo el alta del nuevo usuario.

Prototipo 1.2: Permitir editar sanitario existente.

En el prototipo 1.2, que permite editar un sanitario ya existente, se han realizado las siguientes inserciones:

- Una primitiva de comunicación EDITAR_SANITARIO, que permite reconocer las solicitudes recibidas por parte del cliente.
- No ha sido necesario la creación de nuevas clases DTO.
- En la clase ServidorHospital, se ha creado el siguiente método:
Método editarSanitario(), que lee del socket el sanitario en formato JSON a editar, con los campos ya modificados, y devuelve OK o NOK en función de si la modificación se ha llevado a cabo correctamente.
- En la clase servidorSanitarios, como en la clase anterior, se ha creado el siguiente método.
Método editarSanitario(SanitarioDTO _sanitario), que solicita a la base de datos que modifique un sanitario existente.
Finalmente, notifica a todos los sanitarios activos la modificación del

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

sanitario.

- En la clase que contiene el acceso a la base de datos, DataBaseControl, ha sido necesaria la implementación del siguiente método.
Método editarSanitario(SanitarioDTO _sanitario), que realiza la ejecución de la consulta Update, devolviendo cierto si se ha llevado a cabo correctamente.

Prototipo 1.4: Permitir inicio de sesión de un sanitario.

El prototipo 1.4, ha sido implementado de forma involuntaria llevando a cabo el prototipo 1.3, que permite el inicio de sesión de un administrador. Como se ha indicado anteriormente, todo aquel dado de alta en el sistema es un usuario, que luego se especializará en sanitario o administrador. De esta forma, el prototipo 1.4 y 1.3 son iguales. El único cambio, es que al hacer Select sobre la tabla Administrador cuando se está buscando un sanitario, no obtendremos nada. Por lo tanto, habrá un campo de usuario denominado “admin”, que lo pondremos a false. De esta forma, cuando le pasemos en JSON al cliente el usuario, comprobará el campo “admin”, y si es false, será sanitario, y por tanto accederá como sanitario.

Prototipo 2: Funcionalidad extra Sanitario.


Este prototipo, está formado por 6 sub-prototipos, y hace referencia a la tareas que puede llevar a cabo un sanitario sobre un paciente.

A continuación, se van a indicar los ficheros modificados o creados en cada sub-prototipo para llevar a cabo la tarea indicada.

Prototipo 2.1: Permitir añadir nuevo paciente.

En el prototipo 2.1, que permite dar de alta un paciente, se han realizado las siguientes inserciones:

- Una primitiva de comunicación NUEVO_PACIENTE, que permite reconocer las solicitudes recibidas por parte del cliente.
- Se ha creado la clase PacienteDTO con sus campos correspondientes.


 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

- En la clase ServidorHospital, se ha creado el siguiente método:
Método añadirPaciente(), que lee del socket el paciente en formato JSON a añadir, y devuelve OK o NOK en función de si la inserción se ha llevado a cabo correctamente.
- En la clase servidorSanitarios, como en la clase anterior, se ha creado el siguiente método.
Método añadirPaciente(PacienteDTO _paciente), que solicita a la base de datos que dé de alta un nuevo paciente.
Finalmente, notifica a todos los sanitarios activos la inserción del paciente.
- En la clase que contiene el acceso a la base de datos, DataBaseControl, ha sido necesaria la implementación del siguiente método.
Método addPaciente(PacienteDTO _paciente), que realiza la ejecución de la consulta Insert, devolviendo cierto si se ha llevado a cabo correctamente.

Prototipo 2.2: Permitir añadir un nuevo episodio a un paciente.

En el prototipo 2.2, que permite añadir un nuevo episodio de atención a un paciente, se han realizado las siguientes inserciones:

- Una primitiva de comunicación NUEVO_EPISODIO, que permite reconocer las solicitudes recibidas por parte del cliente.
- Se ha creado la clase EpisodioAtencionDTO con sus campos correspondientes.
- En la clase ServidorHospital, se ha creado el siguiente método:
Método nuevoEpisodio(), que lee del socket el episodio de atención del paciente en formato JSON a añadir, y devuelve OK o NOK en función de si la inserción se ha llevado a cabo correctamente.
- En la clase servidorSanitarios, como en la clase anterior, se ha creado el siguiente método.
Método nuevoEpisodio(EpisodioAtencionDTO _episodio), que solicita a la base de datos que registre el nuevo episodio de atención asociado al paciente.
Finalmente, notifica a todos los sanitarios activos la inserción del episodio.


 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

- En la clase que contiene el acceso a la base de datos, DataBaseControl, ha sido necesaria la implementación del siguiente método.
Método nuevoEpisodio(EpisodioAtencionDTO episodio), que realiza la ejecución de la consulta Insert, devolviendo cierto si se ha llevado a cabo correctamente.

Prototipo 2.3: Permitir añadir una nueva cita a un paciente.

En el prototipo 2.3, que permite añadir una nueva cita a un paciente, se han realizado las siguientes inserciones:


- Una primitiva de comunicación NUEVA_CITA, que permite reconocer las solicitudes recibidas por parte del cliente.
- Se ha creado la clase CitaDTO con sus campos correspondientes.
- En la clase ServidorHospital, se ha creado el siguiente método:
Método nuevaCita(), que lee del socket cita del paciente en formato JSON a añadir, y devuelve OK o NOK en función de si la inserción se ha llevado a cabo correctamente.
- En la clase servidorSanitarios, como en la clase anterior, se ha creado el siguiente método.
Método nuevaCita(CitaDTO _cita), que solicita a la base de datos que registre la nueva cita asociada al paciente y a un sanitario.
Finalmente, notifica a todos los sanitarios activos la inserción de la nueva cita.
- En la clase que contiene el acceso a la base de datos, DataBaseControl, ha sido necesaria la implementación del siguiente método.
Método nuevaCita(CitaDTO _cita), que realiza la ejecución de la consulta Insert, devolviendo cierto si se ha llevado a cabo correctamente.

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

Prototipo 2.4: Permitir añadir un nuevo medicamento a la receta de un paciente.

En el prototipo 2.4, que permite añadir un nuevo medicamento a la receta de un paciente, se han realizado las siguientes inserciones:

- Una primitiva de comunicación `NUEVO_MEDICAMENTO_PACIENTE`, que permite reconocer las solicitudes recibidas por parte del cliente.
- Se han creado las clases `MedicamentoDTO` (hace referencia a los medicamentos dados de alta en el sistema) y `MedicamentoPacienteDTO` (hace referencia al medicamento en la receta de un paciente) con sus campos correspondientes.
- En la clase `ServidorHospital`, se ha creado el siguiente método:
Método `nuevoMedicamentoPaciente()`, que lee del socket un `MedicamentoPaciente` en formato JSON a añadir, y devuelve OK o NOK en función de si la inserción se ha llevado a cabo correctamente.
- En la clase `servidorSanitarios`, como en la clase anterior, se ha creado el siguiente método.
Método `añadirMedicamento(MedicamentoPacienteDTO _medicamento)`, que solicita a la base de datos que registre un medicamento en la receta de un paciente.
Finalmente, notifica a todos los sanitarios activos la inserción de dicho medicamento.
- En la clase que contiene el acceso a la base de datos, `DataBaseControl`, ha sido necesaria la implementación del siguiente método.
Método `añadirMedicamentoAPaciente (MedicamentoPacienteDTO _medicamento)`, que realiza la ejecución de la consulta Insert, devolviendo cierto si se ha llevado a cabo correctamente.

 Universidad Zaragoza	Historia Clínica Especificación de prototipos	Pablo Doñate Adnana Dragut
--	--	---

Prototipo 2.5: Permitir añadir una nueva vacuna a un paciente.

Prototipo 2.6: Permitir obtener la historia completa de un paciente.