Grupo
Calidad

Alberto López Castilla
Ricardo Acedo de Talavera
Universidad Pablo de Olavide
2014-2015

Contenido

1 Descripción del problema a resolver

El proyecto a desarrollar por nuestro grupo consistirá en un software de gestión de reservas para un club de padel. Se tomará un ejemplo de un club realista en el que no sólo se puede alquilar una pista para jugar sino también material. El material puede ser pelotas, palas o incluso luz para jugar por la noche. Además, el club ofertará clases de padel para los socios, por lo que tendrá que gestionar a sus profesores. En el club también se organizarán torneos, como una especie de liga entre los socios.

El sistema también mantendrá un control sobre los socios y las reservas que estos efectúen. El sistema limitará de cierta forma las reservas para evitar que un conjunto de socios puedan reservar de forma abusiva las instalaciones del club.

También se tendrá en cuenta que el material del club siempre será almacenado en el club, y no podrá ser reservado a no ser que se tenga una pista reservada con anterioridad.

Nuestra intención con este software es automatizar una gestión tediosa en papel de forma rápida. La aplicación será de consola y estará programada en Python.

2 Requerimientos y especificación del proyecto

El sistema debe cumplir los siguientes requisitos:

- El sistema debe permitir realizar las operaciones CRUD sobre todas las entidades (socio, reserva, alquiler, profesor, torneo).
- El sistema debe permitir a los socios del club reservar las instalaciones deportivas (teniendo control sobre las reservas antiguas y las pistas reservadas para el torneo).
- El sistema debe permitir a los socios alquilar material deportivo del club.
- El sistema debe permitir cancelar la reserva de instalaciones deportivas.
- El sistema debe permitir devolver el material deportivo del club.
- El sistema debe gestionar el calendario para los torneos.
- El sistema debe permitir a los socios apuntarse a torneos.
- El sistema debe permitir a los socios apuntarse a clases prácticas.

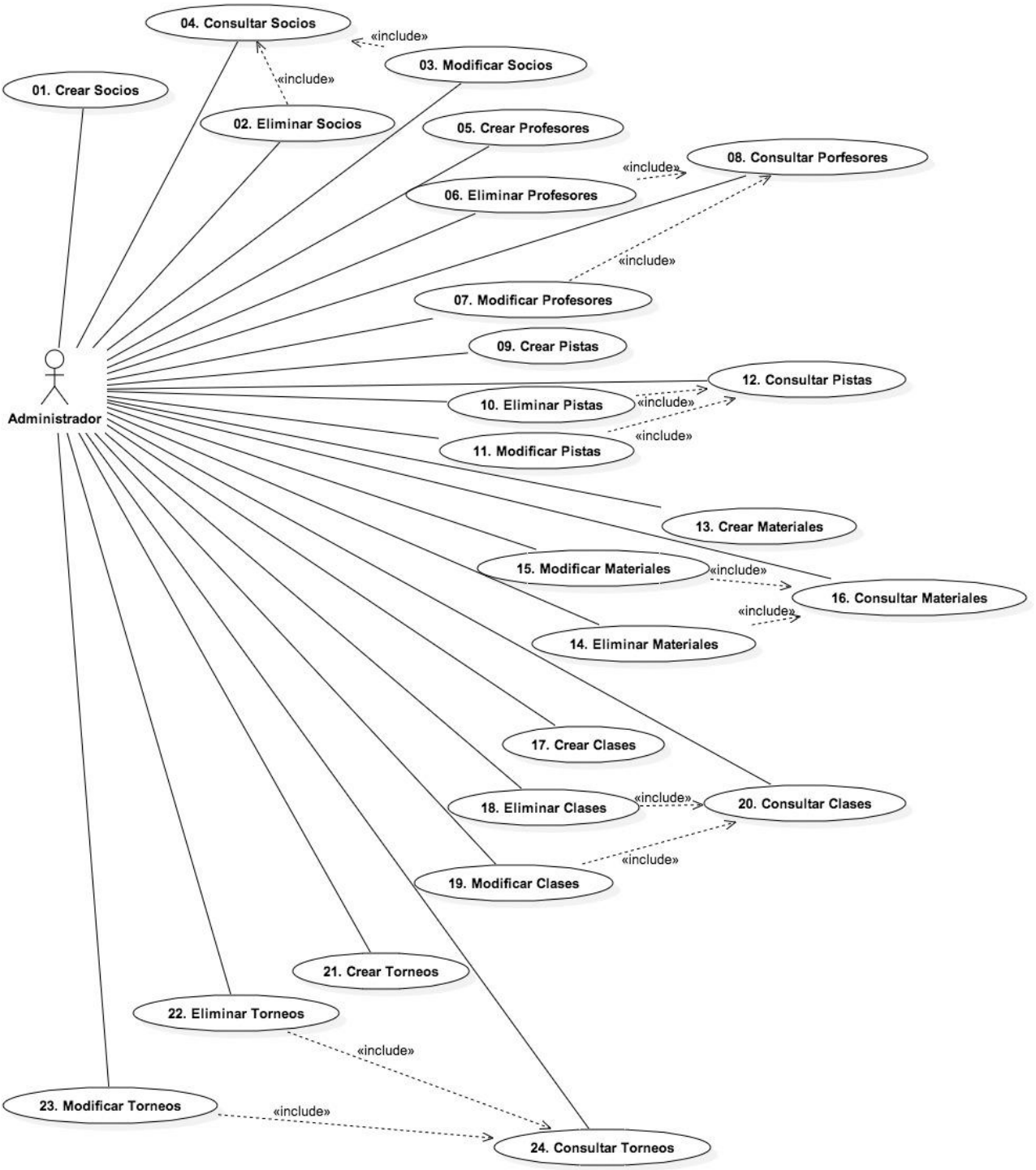
2.1 Especificación de Casos de Uso

2.1.1 Identificación y definición de Actores

Los actores identificados es únicamente el administrador al ser una aplicación de consola.

Actores	Descripción
ACTOR01: Administrador	Es el administrador del sistema en general, que lleva acabo las operaciones de administración de todo el club.

2.1.2 Diagrama de Casos de Uso



2.1.3 Especificación de Casos de uso

CU-01	Crear Socios.	
Descripción	El sistema debe permitir crear nuevos socios.	
Actores	Administrador.	
Precondición		
Flujo Normal	**PASO**	**ACCIÓN**
	1	El actor solicita al sistema crear un socio.
	2	El sistema pide al actor los datos relativos a ese nuevo socio.
	3	El usuario introduce los datos requeridos.
	4	El sistema comprueba que los datos son correctos.
	5	El sistema comprueba que no haya ningún socio igual.
	6	El sistema muestra un mensaje de que se ha creado un nuevo socio.
	7	Finaliza el caso de uso.
Flujos Alternativos	**PASO**	**ACCIÓN**
	4	El sistema comprueba que los datos no son correctos.
	5	El sistema informa al actor de que hay datos incorrectos.
	6	Vuelve al paso 2 del flujo normal.
Flujos Alternativos	**PASO**	**ACCIÓN**
	5	El sistema comprueba que hay un socio igual.
	6	El sistema muestra un mensaje de error y cancela la creación de un nuevo socio.
	7	Vuelve al paso 7 del flujo normal.
Postcondición	Se crea un nuevo socio en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de crear un nuevo socio.	

CU-02	Eliminar Socios.	
Descripción	El sistema debe permitir dar de baja a los socios.	
Actores	Administrador.	
Precondición		
Flujo Normal	**PASO**	**ACCIÓN**
	1	El actor inicia dar de baja a un socio.
	2	El sistema inicia CU-04: Consultar Socios.
	3	El actor selecciona el socio y escoge eliminarlo.
	4	El sistema pide confirmar la eliminación.
	5	El actor confirma.
	6	El sistema elimina el socio seleccionado de la base de datos y muestra un mensaje de finalizado.
	7	Finaliza el caso de uso.
Flujos Alternativos	**PASO**	**ACCIÓN**
	2	El sistema inicia CU-04: Consultar Socios y no encuentra socio registrado.
	3	El sistema muestra un mensaje informativo y vuelve al paso 7 del flujo normal.

Postcondicion	Se elimina un socio en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.
Observaciones	En cualquier momento el actor podrá cancelar la acción de eliminar un socio.

CU-03	Modificar Socios.	
Descripción	El sistema debe permitir modificar los socios.	
Actores	Administrador.	
Precondición		
Flujo Normal	**PASO**	**ACCIÓN**
	1	El actor inicia modificar un socio.
	2	El sistema inicia CU-04: Consultar Socios.
	3	El actor selecciona el socio.
	4	El sistema muestra los datos a modificar del socio seleccionado.
	5	El actor los modifica.
	6	El sistema comprueba que los datos son correctos.
	7	El sistema modifica los datos en la base de datos y muestra un mensaje de finalizado.
	8	Finaliza el caso de uso.
Flujos Alternativos	**PASO**	**ACCIÓN**
	2	El sistema inicia CU-04: Consultar Socios y no encuentra socio registrado.
	3	El sistema muestra un mensaje informativo y vuelve al paso 8 del flujo normal.
Flujos Alternativos	**PASO**	**ACCIÓN**
	6	El sistema comprueba que los datos son incorrectos.
	7	El sistema muestra un mensaje de error.
	8	Vuelve al paso 5 del flujo normal.
Postcondicion	Se modifica un socio en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de modificar un socio.	

CU-04	Consultar Socios.	
Descripción	El sistema debe permitir consultar los socios.	
Actores	Administrador.	
Precondición		
Flujo Normal	**PASO**	**ACCIÓN**
	1	El actor inicia consultar socios.
	2	El sistema muestra la lista de socios que están dados de alta en el sistema.
	3	Si el actor desea buscar un socio por el identificador:
	3.1	El sistema solicita el identificador del socio.
	3.2	El actor introduce el identificador.
	3.3	El sistema comprueba que el dato es correcto.
	3.4	El sistema muestra los datos del socio solicitado.
	4	Finaliza el caso de uso.
Flujos Alternativos	**PASO**	**ACCIÓN**

	2	El sistema muestra una lista vacía porque no hay datos de socios en la base de datos.
	3	Vuelve al paso 4 del flujo normal.
Flujos Alternativos	**PASO**	**ACCIÓN**
	3.3	El sistema comprueba que el dato introducido es erróneo.
	3.4	El sistema muestra un error y vuelve al paso 2 del flujo normal.
Flujos Alternativos	**PASO**	**ACCIÓN**
	3.3	El sistema comprueba que el dato introducido es correcto pero no existe el socio.
	3.4	El sistema muestra una lista vacía y vuelve al paso 4 del flujo normal.
Postcondicion	--	
Observaciones	--	

CU-05	**Crear Profesores.**	
Descripción	El sistema debe permitir crear nuevos profesores.	
Actores	Administrador.	
Precondición		
Flujo Normal	**PASO**	**ACCIÓN**
	1	El actor solicita al sistema crear un profesor.
	2	El sistema pide al actor los datos relativos a ese nuevo profesor.
	3	El usuario introduce los datos requeridos.
	4	El sistema comprueba que los datos son correctos.
	5	El sistema comprueba que no haya ningún profesor igual.
	6	El sistema muestra un mensaje de que se ha creado un nuevo profesor.
	7	Finaliza el caso de uso.
Flujos Alternativos	**PASO**	**ACCIÓN**
	4	El sistema comprueba que los datos no son correctos.
	5	El sistema informa al actor de que hay datos incorrectos.
	6	Vuelve al paso 2 del flujo normal.
Flujos Alternativos	**PASO**	**ACCIÓN**
	5	El sistema comprueba que hay un profesor igual.
	6	El sistema muestra un mensaje de error y cancela la creación de un nuevo profesor.
	7	Vuelve al paso 7 del flujo normal.
Postcondición	Se crea un nuevo profesor en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de crear un nuevo profesor.	

CU-06	**Eliminar Profesores.**

Descripción	El sistema debe permitir dar de baja a los profesores.	
Actores	Administrador.	
Precondición		
Flujo Normal	**PASO**	**ACCIÓN**
	1	El actor inicia dar de baja a un profesor.
	2	El sistema inicia CU-08: Consultar Profesores.
	3	El actor selecciona el profesor y escoge eliminarlo.
	4	El sistema pide confirmar la eliminación.
	5	El actor confirma.
	6	El sistema elimina el profesor seleccionado de la base de datos y muestra un mensaje de finalizado.
	7	Finaliza el caso de uso.
Flujos Alternativos	**PASO**	**ACCIÓN**
	2	El sistema inicia CU-08: Consultar Profesores y no encuentra profesor registrado.
	3	El sistema muestra un mensaje informativo y vuelve al paso 7 del flujo normal.
Postcondicion	Se elimina un profesor en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de eliminar un profesor.	

CU-03	Modificar Profesores.	
Descripción	El sistema debe permitir modificar los profesores.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia modificar un profesor.
	2	El sistema inicia CU-08: Consultar Profesores.
	3	El actor selecciona el profesor.
	4	El sistema muestra los datos a modificar del profesor seleccionado.
	5	El actor los modifica.
	6	El sistema comprueba que los datos son correctos.
	7	El sistema modifica los datos en la base de datos y muestra un mensaje de finalizado.
	8	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema inicia CU-08: Consultar Profesores y no encuentra profesor registrado.
	3	El sistema muestra un mensaje informativo y vuelve al paso 8 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	6	El sistema comprueba que los datos son incorrectos.
	7	El sistema muestra un mensaje de error.
	8	Vuelve al paso 5 del flujo normal.
Postcondicion	Se modifica un profesor en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de	

	modificar un profesor.

CU-08	Consultar Profesores.	
Descripción	El sistema debe permitir consultar los profesores.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia consultar profesores.
	2	El sistema muestra la lista de profesores que están dados de alta en el sistema.
	3	Si el actor desea buscar un profesor por el identificador:
	3.1	El sistema solicita el identificador del profesor.
	3.2	El actor introduce el identificador.
	3.3	El sistema comprueba que el dato es correcto.
	3.4	El sistema muestra los datos del profesor solicitado.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay datos de profesores en la base de datos.
	3	Vuelve al paso 4 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	3.3	El sistema comprueba que el dato introducido es erróneo.
	3.4	El sistema muestra un error y vuelve al paso 2 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	3.3	El sistema comprueba que el dato introducido es correcto pero no existe el profesor.
	3.4	El sistema muestra una lista vacía y vuelve al paso 4 del flujo normal.
Postcondicion	--	
Observaciones	--	

CU-09	Crear Pista.	
Descripción	El sistema debe permitir crear nuevas pistas.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor solicita al sistema crear una pista.
	2	El sistema pide al actor los datos relativos a esa nueva pista.
	3	El usuario introduce los datos requeridos.
	4	El sistema comprueba que los datos son correctos.
	5	El sistema comprueba que no haya ninguna pista igual.
	6	El sistema muestra un mensaje de que se ha creado una nueva pista.
	7	Finaliza el caso de uso.

Flujos Alternativos	PASO	ACCIÓN
	4	El sistema comprueba que los datos no son correctos.
	5	El sistema informa al actor de que hay datos incorrectos.
	6	Vuelve al paso 2 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	5	El sistema comprueba que hay una pista igual.
	6	El sistema muestra un mensaje de error y cancela la creación de una nueva pista.
	7	Vuelve al paso 7 del flujo normal.
Postcondición	Se crea una nueva pista en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de crear una nueva pista.	

CU-10	Eliminar Pistas.	
Descripción	El sistema debe permitir dar de baja a las pistas.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia dar de baja a una pista.
	2	El sistema inicia CU-12: Consultar Pistas.
	3	El actor selecciona la pista y escoge eliminarla.
	4	El sistema pide confirmar la eliminación.
	5	El actor confirma.
	6	El sistema elimina la pista seleccionada de la base de datos y muestra un mensaje de finalizado.
	7	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema inicia CU-12: Consultar Pistas y no encuentra pista registrada.
	3	El sistema muestra un mensaje informativo y vuelve al paso 7 del flujo normal.
Postcondicion	Se elimina una pista en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de eliminar una pista.	

CU-11	Modificar Pistas.	
Descripción	El sistema debe permitir modificar las pistas.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia modificar una pista.
	2	El sistema inicia CU-12: Consultar Pistas.
	3	El actor selecciona la pista.

	4	El sistema muestra los datos a modificar de la pista seleccionada.
	5	El actor los modifica.
	6	El sistema comprueba que los datos son correctos.
	7	El sistema modifica los datos en la base de datos y muestra un mensaje de finalizado.
	8	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema inicia CU-12: Consultar Pistas y no encuentra pista registrada.
	3	El sistema muestra un mensaje informativo y vuelve al paso 8 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	6	El sistema comprueba que los datos son incorrectos.
	7	El sistema muestra un mensaje de error.
	8	Vuelve al paso 5 del flujo normal.
Postcondicion	Se modifica una pista en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de modificar una pista.	

CU-12	Consultar Pistas.	
Descripción	El sistema debe permitir consultar las pistas.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia consultar pistas.
	2	El sistema muestra la lista de pistas que están dadas de alta en el sistema.
	3	Si el actor desea buscar una pista por el identificador:
	3.1	El sistema solicita el identificador de la pista.
	3.2	El actor introduce el identificador.
	3.3	El sistema comprueba que el dato es correcto.
	3.4	El sistema muestra los datos de la pista solicitada.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay datos de pistas en la base de datos.
	3	Vuelve al paso 4 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	3.3	El sistema comprueba que el dato introducido es erróneo.
	3.4	El sistema muestra un error y vuelve al paso 2 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	3.3	El sistema comprueba que el dato introducido es correcto pero no existe la pista.
	3.4	Muestra una lista vacía y vuelve al paso 4 del flujo normal.

Postcondicion	--
Observaciones	--

CU-13	Crear Material.	
Descripción	El sistema debe permitir crear nuevos materiales.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor solicita al sistema crear un material.
	2	El sistema pide al actor los datos relativos a ese nuevo material.
	3	El usuario introduce los datos requeridos.
	4	El sistema comprueba que los datos son correctos.
	5	El sistema comprueba que no haya ningún material igual.
	6	El sistema muestra un mensaje de que se ha creado un nuevo material.
	7	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	4	El sistema comprueba que los datos no son correctos.
	5	El sistema informa al actor de que hay datos incorrectos.
	6	Vuelve al paso 2 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	5	El sistema comprueba que hay un material igual.
	6	El sistema muestra un mensaje de error y cancela la creación de un nuevo material.
	7	Vuelve al paso 7 del flujo normal.
Postcondición	Se crea un nuevo material en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de crear un nuevo material.	

CU-14	Eliminar Materiales.	
Descripción	El sistema debe permitir dar de baja a los materiales.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia dar de baja a un material.
	2	El sistema inicia CU-16: Consultar Materiales.
	3	El actor selecciona el material y escoge eliminarlo.
	4	El sistema pide confirmar la eliminación.
	5	El actor confirma.
	6	El sistema elimina el material seleccionado de la base de datos y muestra un mensaje de finalizado.
	7	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN

	2	El sistema inicia CU-16: Consultar Materiales y no encuentra material registrado.
	3	El sistema muestra un mensaje informativo y vuelve al paso 7 del flujo normal.
Postcondicion	Se elimina un material en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de eliminar un material.	

CU-15	Modificar Materiales.	
Descripción	El sistema debe permitir modificar los materiales.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia modificar un material.
	2	El sistema inicia CU-16: Consultar Materiales.
	3	El actor selecciona el material.
	4	El sistema muestra los datos a modificar del material seleccionado.
	5	El actor los modifica.
	6	El sistema comprueba que los datos son correctos.
	7	El sistema modifica los datos en la base de datos y muestra un mensaje de finalizado.
	8	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema inicia CU-16: Consultar Materiales y no encuentra material registrado.
	3	El sistema muestra un mensaje informativo y vuelve al paso 8 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	6	El sistema comprueba que los datos son incorrectos.
	7	El sistema muestra un mensaje de error.
	8	Vuelve al paso 5 del flujo normal.
Postcondicion	Se modifica un material en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de modificar un material.	

CU-16	Consultar Materiales.	
Descripción	El sistema debe permitir consultar los materiales.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia consultar materiales.
	2	El sistema muestra la lista de materiales que están dados de alta en el sistema.
	3	Si el actor desea buscar un material por el

		identificador:
	3.1	El sistema solicita el identificador del material.
	3.2	El actor introduce el identificador.
	3.3	El sistema comprueba que el dato es correcto.
	3.4	El sistema muestra los datos del material solicitado.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay datos de materiales en la base de datos.
	3	Vuelve al paso 4 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	3.3	El sistema comprueba que el dato introducido es erróneo.
	3.4	El sistema muestra un error y vuelve al paso 2 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	3.3	El sistema comprueba que el dato introducido es correcto pero no existe el material.
	3.4	El sistema muestra una lista vacía y vuelve al paso 4 del flujo normal.
Postcondicion	--	
Observaciones	--	

CU-17	Crear Clases.	
Descripción	El sistema debe permitir crear nuevas clases.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor solicita al sistema crear una clase.
	2	El sistema pide al actor los datos relativos a esa nueva clase.
	3	El usuario introduce los datos requeridos.
	4	El sistema comprueba que los datos son correctos.
	5	El sistema comprueba que no haya ninguna clase igual.
	6	El sistema muestra un mensaje de que se ha creado una nueva clase.
	7	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	4	El sistema comprueba que los datos no son correctos.
	5	El sistema informa al actor de que hay datos incorrectos.
	6	Vuelve al paso 2 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	5	El sistema comprueba que hay una clase igual.
	6	El sistema muestra un mensaje de error y cancela la creación de una nueva clase.
	7	Vuelve al paso 7 del flujo normal.
Postcondición	Se crea una nueva clase en la base de datos, a no ser que se	

	cancele la acción y se dejará la base de datos como estaba al inicio.
Observaciones	En cualquier momento el actor podrá cancelar la acción de crear una nueva clase.

CU-18	Eliminar Clases.	
Descripción	El sistema debe permitir dar de baja a las clases.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia dar de baja a una clase.
	2	El sistema inicia CU-20: Consultar Clases.
	3	El actor selecciona la clase y escoge eliminarla.
	4	El sistema pide confirmar la eliminación.
	5	El actor confirma.
	6	El sistema elimina la clase seleccionada de la base de datos y muestra un mensaje de finalizado.
	7	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema inicia CU-20: Consultar Clases y no encuentra clase registrada.
	3	El sistema muestra un mensaje informativo y vuelve al paso 7 del flujo normal.
Postcondicion	Se elimina una clase en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de eliminar una clase.	

CU-19	Modificar Clases.	
Descripción	El sistema debe permitir modificar las clases.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia modificar una clase.
	2	El sistema inicia CU-20: Consultar Clases.
	3	El actor selecciona la clase.
	4	El sistema muestra los datos a modificar de la clase seleccionada.
	5	El actor los modifica.
	6	El sistema comprueba que los datos son correctos.
	7	El sistema modifica los datos en la base de datos y muestra un mensaje de finalizado.
	8	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema inicia CU-20: Consultar Clases y no encuentra clase registrada.
	3	El sistema muestra un mensaje informativo y vuelve al paso 8 del flujo normal.

Flujos Alternativos	PASO	ACCIÓN
	6	El sistema comprueba que los datos son incorrectos.
	7	El sistema muestra un mensaje de error.
	8	Vuelve al paso 5 del flujo normal.
Postcondicion	Se modifica una clase en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de modificar una clase.	

CU-20	Consultar Clases.	
Descripción	El sistema debe permitir consultar las clases.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia consultar clases.
	2	El sistema muestra la lista de clases que están dadas de alta en el sistema.
	3	Si el actor desea buscar una clase por el identificador:
	3.1	El sistema solicita el identificador de la clase.
	3.2	El actor introduce el identificador.
	3.3	El sistema comprueba que el dato es correcto.
	3.4	El sistema muestra los datos de la clase solicitada.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay datos de clases en la base de datos.
	3	Vuelve al paso 4 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	3.3	El sistema comprueba que el dato introducido es erróneo.
	3.4	El sistema muestra un error y vuelve al paso 2 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	3.3	El sistema comprueba que el dato introducido es correcto pero no existe la clase.
	3.4	Muestra una lista vacía y vuelve al paso 4 del flujo normal.
Postcondicion	--	
Observaciones	--	

CU-21	Crear Torneo.	
Descripción	El sistema debe permitir crear nuevos torneos.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor solicita al sistema crear un torneo.
	2	El sistema pide al actor los datos relativos a ese nuevo torneo.

	3	El usuario introduce los datos requeridos.
	4	El sistema comprueba que los datos son correctos.
	5	El sistema comprueba que no haya ningún torneo igual.
	6	El sistema muestra un mensaje de que se ha creado un nuevo torneo.
	7	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	4	El sistema comprueba que los datos no son correctos.
	5	El sistema informa al actor de que hay datos incorrectos.
	6	Vuelve al paso 2 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	5	El sistema comprueba que hay un torneo igual.
	6	El sistema muestra un mensaje de error y cancela la creación de un nuevo torneo.
	7	Vuelve al paso 7 del flujo normal.
Postcondición	Se crea un nuevo torneo en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de crear un nuevo torneo.	

CU-22	Eliminar Torneos.	
Descripción	El sistema debe permitir dar de baja a los torneos.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia dar de baja a un torneo.
	2	El sistema inicia CU-24: Consultar Torneos.
	3	El actor selecciona el torneo y escoge eliminarlo.
	4	El sistema pide confirmar la eliminación.
	5	El actor confirma.
	6	El sistema elimina el torneo seleccionado de la base de datos y muestra un mensaje de finalizado.
	7	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema inicia CU-24: Consultar Torneos y no encuentra torneo registrado.
	3	El sistema muestra un mensaje informativo y vuelve al paso 7 del flujo normal.
Postcondicion	Se elimina un torneo en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de eliminar un torneo.	

CU-23	Modificar Torneos.
Descripción	El sistema debe permitir modificar los torneos.

Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia modificar un torneo.
	2	El sistema inicia CU-24: Consultar Torneos.
	3	El actor selecciona el torneo.
	4	El sistema muestra los datos a modificar del torneo seleccionado.
	5	El actor los modifica.
	6	El sistema comprueba que los datos son correctos.
	7	El sistema modifica los datos en la base de datos y muestra un mensaje de finalizado.
	8	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema inicia CU-24: Consultar Torneos y no encuentra torneo registrado.
	3	El sistema muestra un mensaje informativo y vuelve al paso 8 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	6	El sistema comprueba que los datos son incorrectos.
	7	El sistema muestra un mensaje de error.
	8	Vuelve al paso 5 del flujo normal.
Postcondicion	Se modifica un torneo en la base de datos, a no ser que se cancele la acción y se dejará la base de datos como estaba al inicio.	
Observaciones	En cualquier momento el actor podrá cancelar la acción de modificar un torneo.	

CU-24	Consultar Torneos.	
Descripción	El sistema debe permitir consultar los torneos.	
Actores	Administrador.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia consultar torneos.
	2	El sistema muestra la lista de torneos que están dados de alta en el sistema.
	3	Si el actor desea buscar un torneo por el identificador:
	3.1	El sistema solicita el identificador del torneo.
	3.2	El actor introduce el identificador.
	3.3	El sistema comprueba que el dato es correcto.
	3.4	El sistema muestra los datos del material solicitado.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay datos de torneos en la base de datos.
	3	Vuelve al paso 4 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	3.3	El sistema comprueba que el dato introducido es erróneo.

	3.4	El sistema muestra un error y vuelve al paso 2 del flujo normal.
Flujos Alternativos	PASO	ACCIÓN
	3.3	El sistema comprueba que el dato introducido es correcto pero no existe el torneo.
	3.4	El sistema muestra una lista vacía y vuelve al paso 4 del flujo normal.
Postcondicion	--	
Observaciones	--	

CU-25	Alquiler Pistas.	
Descripción	El sistema debe permitir alquilar las pistas.	
Actores	Socio.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia alquiler de pistas.
	2	El sistema muestra la lista de pistas que están libres día/hora.
	3	El actor selecciona la pista que quiere alquilar día/hora
	3.1	El sistema solicita al actor que confirme el alquiler mostrando un resumen del alquiler.
	3.2	El actor confirma el alquiler.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay datos de pistas en la base de datos.
	3	Vuelve al paso 4 del flujo normal.
Postcondicion	--	
Observaciones	El actor puede cancelar en cualquier momento el alquiler de pista.	

CU-26	Cancelación Pistas.	
Descripción	El sistema debe permitir cancelar el alquiler de pistas.	
Actores	Socio.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia cancelar el alquiler de pistas.
	2	El sistema muestra la lista de pistas que tiene alquilada el actor
	3	El actor selecciona la pista que quiere cancelar el alquiler.
	3.1	El sistema solicita al actor que confirme la cancelación.
	3.2	El actor confirma la cancelación.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay datos de pistas alquiladas.
	3	Vuelve al paso 4 del flujo normal.

Postcondicion	--
Observaciones	El actor puede cancelar en cualquier momento la cancelación alquiler de pista.

CU-27	Alquiler Material.	
Descripción	El sistema debe permitir alquilar los materiales.	
Actores	Socio.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia alquiler de materiales.
	2	El sistema muestra la lista de materiales que están libres día/hora.
	3	El actor selecciona el material que quiere alquilar día/hora
	3.1	El sistema solicita al actor que confirme el alquiler mostrando un resumen del alquiler.
	3.2	El actor confirma el alquiler.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay datos de materiales en la base de datos.
	3	Vuelve al paso 4 del flujo normal.
Postcondicion	--	
Observaciones	El actor puede cancelar en cualquier momento el alquiler de materiales.	

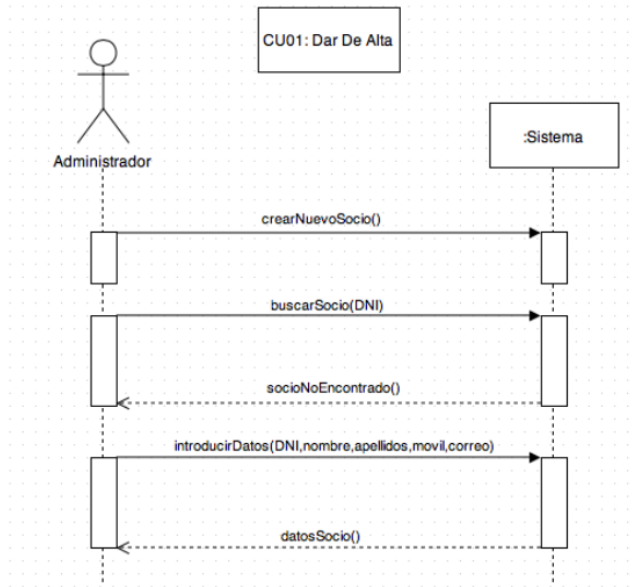
CU-28	Devolver Materiales.	
Descripción	El sistema debe permitir devolver el alquiler de materiales.	
Actores	Socio.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia devolver el alquiler de materiales.
	2	El sistema muestra la lista de materiales que tiene alquilado el actor
	3	El actor selecciona el material que quiere devolver.
	3.1	El sistema solicita al actor que confirme la devolución.
	3.2	El actor confirma la cancelación.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay datos de materiales alquilados.
	3	Vuelve al paso 4 del flujo normal.
Postcondicion	--	
Observaciones	El actor puede cancelar en cualquier momento la cancelación alquiler de materiales.	

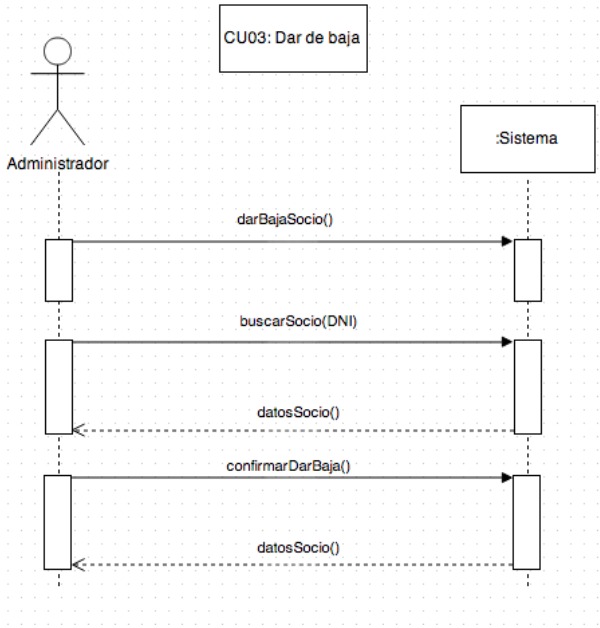
CU-29	Apuntarse a Clases.	
Descripción	El sistema debe permitir apuntarse a clases.	
Actores	Socio.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia apuntarse a clases
	2	El sistema muestra la lista de clases que hay.
	3	El actor selecciona la clase a la que quiere apuntarse.
	3.1	El sistema solicita al actor que confirme que se ha apuntado.
	3.2	El actor confirma que se ha apuntado.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay clases para impartir.
	3	Vuelve al paso 4 del flujo normal.
Postcondicion	--	
Observaciones	El actor puede cancelar en cualquier momento la cancelación de apuntarse a clases.	

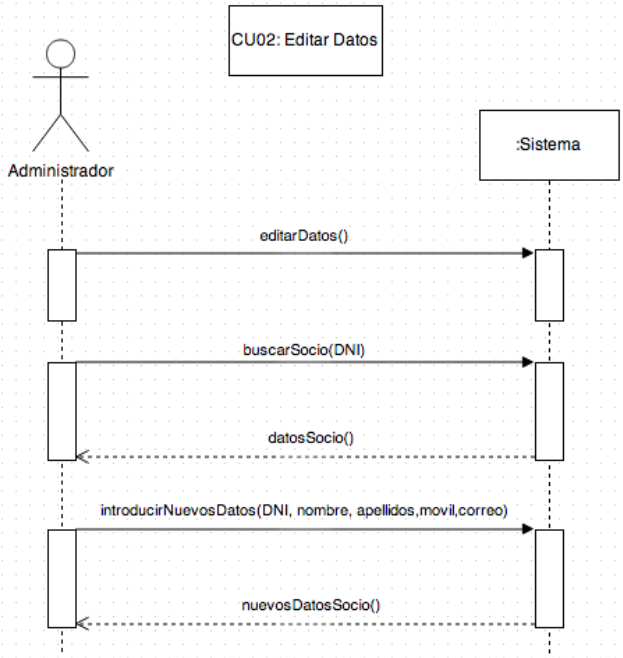
CU-30	Apuntarse a Torneos.	
Descripción	El sistema debe permitir apuntarse a torneos.	
Actores	Socio.	
Precondición		
Flujo Normal	PASO	ACCIÓN
	1	El actor inicia apuntarse a un torneo
	2	El sistema muestra la lista de torneos que hay.
	3	El actor selecciona el torneo al que quiere apuntarse.
	3.1	El sistema solicita al actor que confirme que se ha apuntado.
	3.2	El actor confirma que se ha apuntado.
	4	Finaliza el caso de uso.
Flujos Alternativos	PASO	ACCIÓN
	2	El sistema muestra una lista vacía porque no hay torneos para jugar.
	3	Vuelve al paso 4 del flujo normal.
Postcondicion	--	
Observaciones	El actor puede cancelar en cualquier momento la cancelación de apuntarse a un torneo.	

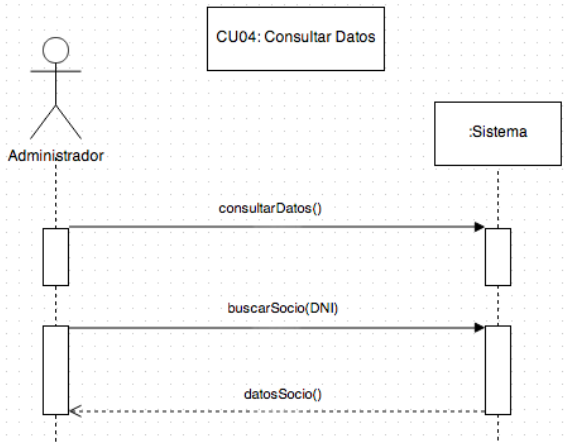
3 Diseño y arquitectura

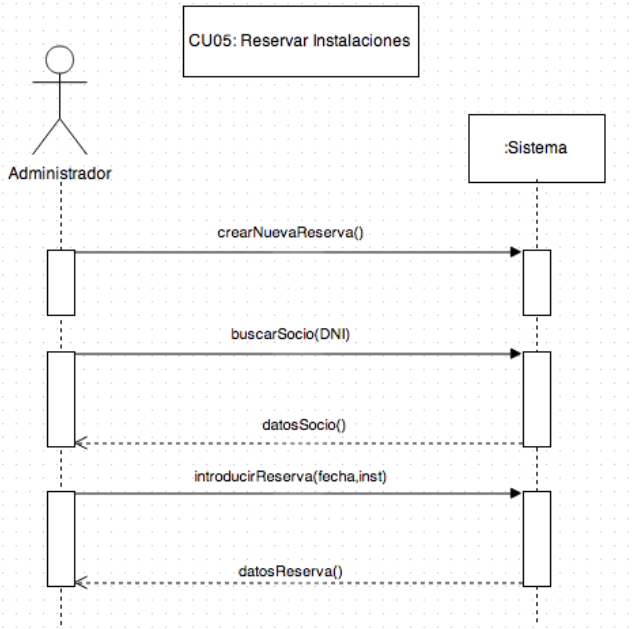
3.1 Diagramas de clase

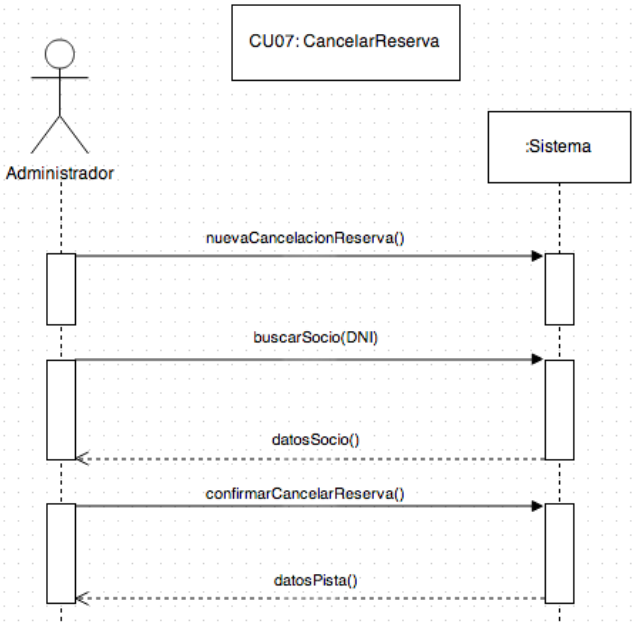


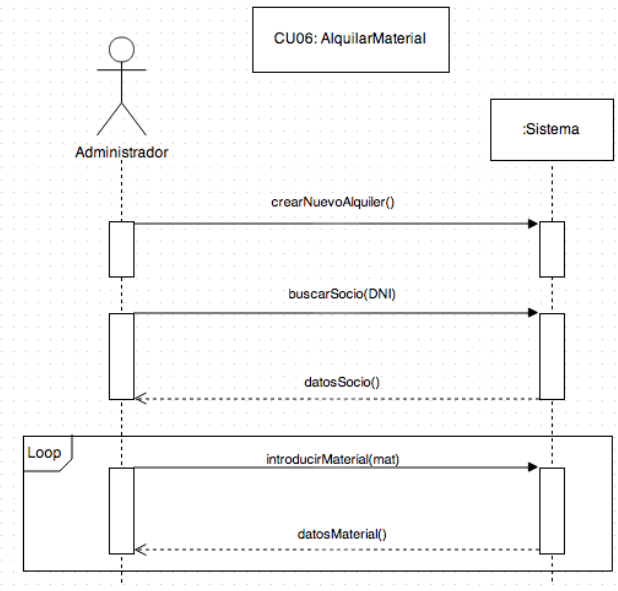


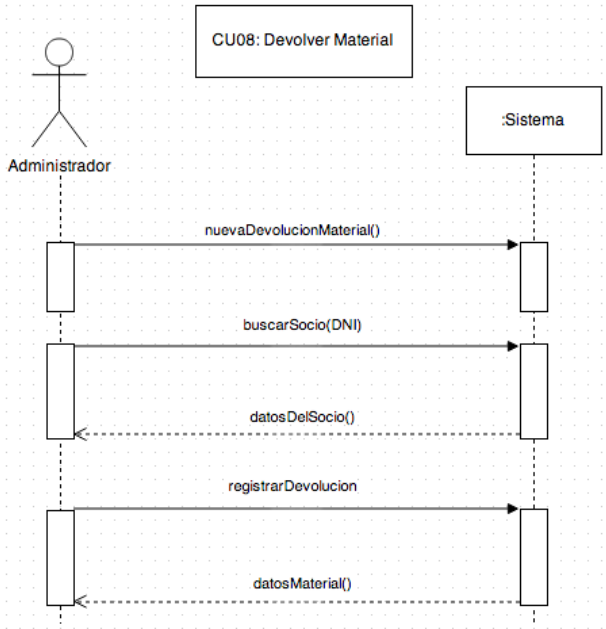





CU05: Reservar Instalaciones
Administrador
:Sistema
crearNuevaReserva()
buscarSocio(DNI)
datosSocio()
introducirReserva(fecha,inst)
datosReserva()


CU07: CancelarReserva
Administrador
:Sistema
nuevaCancelacionReserva()
buscarSocio(DNI)
datosSocio()
confirmarCancelarReserva()
datosPista()


CU06: AlquilarMaterial
Administrador
:Sistema
crearNuevoAlquiler()
buscarSocio(DNI)
datosSocio()
Loop
introducirMaterial(mat)
datosMaterial()


CU08: Devolver Material
Administrador
:Sistema
nuevaDevolucionMaterial()
buscarSocio(DNI)
datosDelSocio()
registrarDevolucion
datosMaterial()

4 Introducción de la calidad del proyecto

Para el desarrollo del proyecto de la asignatura "Calidad" lo primero que hicimos fue crear un repositorio en github con los miembros del grupo como colaboradores. Dicho repositorio puede ser encontrado en la siguiente dirección:

https://github.com/grupocalidad/Proyecto

Después de ello, empezamos todos a desarrollar una primera versión tal y como se indicaba en las instrucciones del proyecto. Cuando esta estuvo lista, nos dispusimos a emplear las herramientas aprendidas a lo largo de la asignatura para ratificar cierto nivel de calidad de nuestro producto final.

5 Métricas v 0.1

Una vez que terminamos de desarrollar la primera versión (Con una etiqueta v0.1 en github) procedimos a utilizar las herramientas que aprendimos en las prácticas de esta asignatura para ver las carencias de nuestro proyecto. A continuación se explican cada uno de los resultados obtenidos.

5.1 Cobertura de los tests

Como es evidente, el coverage de nuestro código fue nulo. Nos centramos únicamente en el desarrollo del código y no hicimos en ese entonces ni un sólo test unitario con el fin de centrarnos únicamente en el desarrollo y de tener una version totalmente funcional lo antes posible.

```
---------- coverage: platform linux2, python 2.7.6-final-0 ----------
Name                                                 Stmts   Miss  Cover
------------------------------------------------------------------------------
/home/alberto/PycharmProjects/Proyecto_calidad/Alquiler       25     25     0%
/home/alberto/PycharmProjects/Proyecto_calidad/Clase          16     16     0%
/home/alberto/PycharmProjects/Proyecto_calidad/Club          120    120     0%
/home/alberto/PycharmProjects/Proyecto_calidad/Conexion      396    396     0%
/home/alberto/PycharmProjects/Proyecto_calidad/Instalacion    17     17     0%
/home/alberto/PycharmProjects/Proyecto_calidad/Main          211    211     0%
/home/alberto/PycharmProjects/Proyecto_calidad/Profesor       17     17     0%
/home/alberto/PycharmProjects/Proyecto_calidad/Reserva        25     25     0%
/home/alberto/PycharmProjects/Proyecto_calidad/Socio          51     51     0%
/home/alberto/PycharmProjects/Proyecto_calidad/Torneo         19     19     0%
/home/alberto/PycharmProjects/Proyecto_calidad/__init__        1      1     0%
------------------------------------------------------------------------------
TOTAL                                                 898    898     0%

============================ 4 passed in 0.91 seconds ============================
```

5.2 Complejidad ciclomática

La complejidad ciclomática inicial fue sorprendentemente buena, el buen diseño había producido un buen encapsulamiento de todas las funciones y había hecho que no se produjera funciones con excesiva complejidad ciclomática (Sólo las más complejas tienen complejidad B)

```
M 6:4 Socio.__init__ - A (2)
  C 5:0 Socio - A (1)
  M 19:4 Socio.__str__ - A (1)
  M 22:4 Socio.get_DNI - A (1)
  M 26:4 Socio.get_nombre - A (1)
M 29:4 Socio.get_apellidos - A (1)
M 33:4 Socio.get_movil - A (1)
M 36:4 Socio.get_correo - A (1)
M 39:4 Socio.get_fechaAlta - A (1)
M 42:4 Socio.get_estado - A (1)
```

```
        M 45:4 Socio.set_DNI - A (1)
        M 48:4 Socio.cambiar_estado - A (1)
        M 51:4 Socio.set_fechaAlta - A (1)
        M 54:4 Socio.set_nombre - A (1)
        M 57:4 Socio.set_apellidos - A (1)
        M 60:4 Socio.set_correo - A (1)
        M 63:4 Socio.set_movil - A (1)
    /home/alberto/PycharmProjects/Proyecto_calidad/Clase.py
        C 3:0 Clase - A (1)
        M 4:4 Clase.__init__ - A (1)
        M 8:4 Clase.get_profesor - A (1)
        M 11:4 Clase.get_reserva - A (1)
        M 14:4 Clase.set_profesor - A (1)
        M 17:4 Clase.set_reserva! - A (1)
        M 20:4 Clase.__str__ - A (1)
    /home/alberto/PycharmProjects/Proyecto_calidad/Alquiler.py
        M 28:4 Alquiler.__str__ - A (2)
        C 3:0 Alquiler - A (1)
        M 4:4 Alquiler.isDevuelto - A (1)
        M 7:4 Alquiler.get_reserva - A (1)
        M 10:4 Alquiler.get_instalaciones - A (1)
        M 13:4 Alquiler.aniadirInstalacion - A (1)
        M 16:4 Alquiler.set_devuelto - A (1)
        M 23:4 Alquiler.__init__ - A (1)
    /home/alberto/PycharmProjects/Proyecto_calidad/Conexion.py
        M 186:4 Conexion.dar_baja_profesor - B (8)
        M 343:4 Conexion.cambiar_socio - B (8)
        M 377:4 Conexion.cambiar_profesor - B (8)
        M 87:4 Conexion.__listar_torneos - B (6)
        M 329:4 Conexion.borrar_clase - B (6)
        M 68:4 Conexion.__listar_alquileres - A (5)
        M 166:4 Conexion.dar_baja_socio - A (5)
        M 222:4 Conexion.sacar_clase - A (5)
        M 253:4 Conexion.sacar_reserva - A (5)
        M 266:4 Conexion.borrar_reserva - A (5)
        M 302:4 Conexion.devolver_alquiler - A (5)
        M 142:4 Conexion.sacar_socio - A (4)
        M 154:4 Conexion.sacar_profesor - A (4)
        M 212:4 Conexion.sacar_instalacion - A (4)
        M 290:4 Conexion.sacar_alquiler - A (4)
        M 432:4 Conexion.sacar_torneo - A (4)
        C 15:0 Conexion - A (3)
        M 17:4 Conexion.__listar_socios - A (3)
        M 26:4 Conexion.__listar_clases - A (3)
        M 37:4 Conexion.__listar_instalaciones - A (3)
        M 47:4 Conexion.__listar_reservas - A (3)
        M 59:4 Conexion.__listar_profesores - A (3)
        M 421:4 Conexion.guardar_torneo_fichero - A (3)
        M 451:4 Conexion.borrar_torneo - A (3)
        M 320:4 Conexion.guardar_alquiler_fichero - A (2)
        M 444:4 Conexion.poner_resultado - A (2)
        M 103:4 Conexion.__init__ - A (1)
        M 112:4 Conexion.guardar_socio - A (1)
        M 126:4 Conexion.guardar_profesor - A (1)
        M 234:4 Conexion.guardar_reserva - A (1)
        M 244:4 Conexion.guardar_clase - A (1)
        M 284:4 Conexion.guardar_alquiler - A (1)
        M 415:4 Conexion.guardar_torneo - A (1)
    /home/alberto/PycharmProjects/Proyecto_calidad/Torneo.py
        C 3:0 Torneo - A (1)
        M 4:4 Torneo.__init__ - A (1)
        M 9:4 Torneo.get_nombre - A (1)
        M 12:4 Torneo.get_socios - A (1)
        M 15:4 Torneo.get_resultados - A (1)
        M 18:4 Torneo.set_resultado - A (1)
        M 25:4 Torneo.__eq__ - A (1)
    /home/alberto/PycharmProjects/Proyecto_calidad/Club.py
        M 118:4 Club.validarDNI - B (6)
        M 104:4 Club.crear_alquiler - A (3)
        M 141:4 Club.validarTelefono - A (3)
        M 148:4 Club.validarFecha - A (3)
        M 40:4 Club.dar_baja_socio - A (2)
        M 68:4 Club.crear_torneo - A (2)
        M 134:4 Club.validarEmail - A (2)
        C 12:0 Club - A (1)
        M 13:4 Club.__init__ - A (1)
        M 16:4 Club.alta_socio - A (1)
        M 21:4 Club.alta_profesor - A (1)
        M 26:4 Club.editar_socio - A (1)
        M 29:4 Club.editar_profesor - A (1)
        M 32:4 Club.obtener_socio - A (1)
        M 36:4 Club.obtener_profesor - A (1)
        M 46:4 Club.dar_baja_profesor - A (1)
        M 49:4 Club.crear_reserva - A (1)
        M 59:4 Club.consultar_reserva - A (1)
        M 64:4 Club.cancelar_reserva - A (1)
        M 77:4 Club.consultar_torneo - A (1)
        M 80:4 Club.introducir_resultado - A (1)
        M 84:4 Club.borrar_torneo - A (1)
        M 88:4 Club.cancelar_clase - A (1)
        M 92:4 Club.obtener_instalacion - A (1)
        M 96:4 Club.obtener_clase - A (1)
        M 100:4 Club.registrar_clase - A (1)
        M 112:4 Club.devolver_alquiler - A (1)
        M 115:4 Club.consultar_alquiler - A (1)
    /home/alberto/PycharmProjects/Proyecto_calidad/Reserva.py
        M 31:4 Reserva.__eq__ - A (2)
        C 2:0 Reserva - A (1)
        M 4:4 Reserva.get_fecha - A (1)
        M 7:4 Reserva.set_fecha - A (1)
        M 10:4 Reserva.get_socio - A (1)
        M 13:4 Reserva.set_socio - A (1)
        M 16:4 Reserva.get_instalacion - A (1)
        M 19:4 Reserva.set_instalacion - A (1)
        M 26:4 Reserva.__init__ - A (1)
        M 34:4 Reserva.__str__ - A (1)
    /home/alberto/PycharmProjects/Proyecto_calidad/Main.py
        F 4:0 pedir_reserva - A (1)
        F 10:0 consultar_reserva - A (1)
    /home/alberto/PycharmProjects/Proyecto_calidad/Instalacion.py
        C 3:0 Instalacion - A (1)
        M 4:4 Instalacion.get_descripcion - A (1)
        M 7:4 Instalacion.get_precio - A (1)
        M 10:4 Instalacion.get_id - A (1)
        M 17:4 Instalacion.__init__ - A (1)
        M 22:4 Instalacion.__str__ - A (1)
    /home/alberto/PycharmProjects/Proyecto_calidad/Profesor.py
```

```
    M 5:4 Profesor.__init__ - A (2)
    C 4:0 Profesor - A (1)
    M 13:4 Profesor.set_salario - A (1)
    M 16:4 Profesor.set_jornada - A (1)
    M 19:4 Profesor.get_salario - A (1)
    M 22:4 Profesor.get_jornada - A (1)

Process finished with exit code 0
```

5.3 Índice de mantenibilidad

Nuestra gran sorpresa fue el índice de mantenibilidad; nos sorprendimos mucho cuando vimos que el modelo de nuestro programa tenía un índice bastante bajo. Esto se debía principalmente a que fue de mayor envergadura al final de lo que nosotros pensábamos que iba a ser.

```
    /home/alberto/PycharmProjects/Proyecto_calidad/Socio.py - A
    /home/alberto/PycharmProjects/Proyecto_calidad/Clase.py - A
    /home/alberto/PycharmProjects/Proyecto_calidad/Alquiler.py - A
    /home/alberto/PycharmProjects/Proyecto_calidad/Conexion.py - B
    /home/alberto/PycharmProjects/Proyecto_calidad/Torneo.py - A
    /home/alberto/PycharmProjects/Proyecto_calidad/Club.py - A
    /home/alberto/PycharmProjects/Proyecto_calidad/Reserva.py - A
    /home/alberto/PycharmProjects/Proyecto_calidad/Main.py - A
    /home/alberto/PycharmProjects/Proyecto_calidad/Instalacion.py - A
    /home/alberto/PycharmProjects/Proyecto_calidad/__init__.py - A
    /home/alberto/PycharmProjects/Proyecto_calidad/Profesor.py – A
```

5.4 Métricas crudas

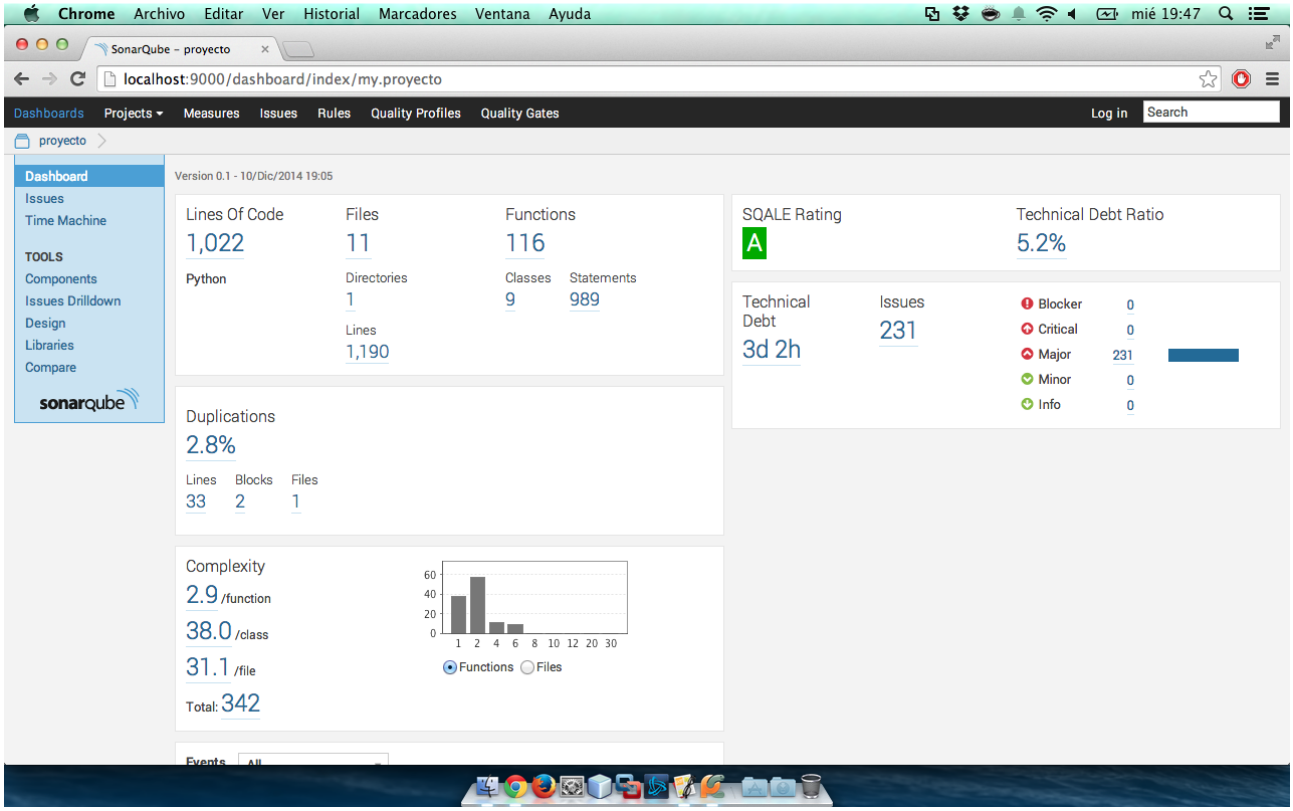
Las métricas crudas también arrojaron resultados un poco pobres, ya que la cantidad de comentarios en el código era muy baja debido, una vez más, a nuestra priorización de obtener una primera versión funcional de nuestro programa. Las métricas arrojadas por la herramienta radon son las siguientes:

```
    /usr/local/bin/radon raw
/home/alberto/PycharmProjects/Proyecto_calidad
    /home/alberto/PycharmProjects/Proyecto_calidad/Socio.py
        LOC: 78
        LLOC: 52
        SLOC: 57
        Comments: 5
        Multi: 0
        Blank: 21
        - Comment Stats
          (C % L): 6%
          (C % S): 9%
          (C + M % L): 6%
    /home/alberto/PycharmProjects/Proyecto_calidad/Clase.py
        LOC: 21
        LLOC: 16
        SLOC: 16
        Comments: 0
        Multi: 0
        Blank: 5
        - Comment Stats
          (C % L): 0%
          (C % S): 0%
          (C + M % L): 0%
    /home/alberto/PycharmProjects/Proyecto_calidad/Alquiler.py
        LOC: 33
        LLOC: 25
        SLOC: 25
        Comments: 0
        Multi: 0
        Blank: 8
        - Comment Stats
          (C % L): 0%
          (C % S): 0%
          (C + M % L): 0%
    /home/alberto/PycharmProjects/Proyecto_calidad/Conexion.py
        LOC: 462
        LLOC: 411
        SLOC: 416
        Comments: 4
        Multi: 0
        Blank: 46
        - Comment Stats
          (C % L): 1%
          (C % S): 1%
          (C + M % L): 1%
    /home/alberto/PycharmProjects/Proyecto_calidad/Torneo.py
        LOC: 26
        LLOC: 20
        SLOC: 19
        Comments: 0
        Multi: 0
        Blank: 7
        - Comment Stats
          (C % L): 0%
```

```
            (C % S): 0%
            (C + M % L): 0%
        /home/alberto/PycharmProjects/Proyecto_calidad/Club
.py
            LOC: 155
            LLOC: 125
            SLOC: 128
            Comments: 4
            Multi: 0
            Blank: 27
            - Comment Stats
                (C % L): 3%
                (C % S): 3%
                (C + M % L): 3%
        /home/alberto/PycharmProjects/Proyecto_calidad/Res
erva.py
            LOC: 35
            LLOC: 25
            SLOC: 25
            Comments: 0
            Multi: 0
            Blank: 10
            - Comment Stats
                (C % L): 0%
                (C % S): 0%
                (C + M % L): 0%
        /home/alberto/PycharmProjects/Proyecto_calidad/Main
.py
            LOC: 230
            LLOC: 216
            SLOC: 216
            Comments: 0
            Multi: 0
            Blank: 14
            - Comment Stats
                (C % L): 0%
                (C % S): 0%
                (C + M % L): 0%
        /home/alberto/PycharmProjects/Proyecto_calidad/Insta
lacion.py
            LOC: 23
            LLOC: 17
            SLOC: 17
            Comments: 0
            Multi: 0
            Blank: 6
            - Comment Stats
                (C % L): 0%
                (C % S): 0%
                (C + M % L): 0%
        /home/alberto/PycharmProjects/Proyecto_calidad/__ini
t__.py
            LOC: 1
            LLOC: 1
            SLOC: 1
            Comments: 0
            Multi: 0
            Blank: 0
            - Comment Stats
                (C % L): 0%
                (C % S): 0%
                (C + M % L): 0%
        /home/alberto/PycharmProjects/Proyecto_calidad/Prof
esor.py
            LOC: 23
            LLOC: 18
            SLOC: 18
            Comments: 0
            Multi: 0
            Blank: 5
            - Comment Stats
                (C % L): 0%
                (C % S): 0%
                (C + M % L): 0%
```

5.5 Duplicidad de código

Para ver la duplicidad de código junto con otras métricas adicionales utilizamos “sonarqube”. Utilizamos las reglas por defecto que trae sonarqube con excepción de un par de ellas que editamos para obtener justo lo que queríamos.



Nos dimos cuenta de que teníamos duplicidad en una clase de una pequeña porción de código.

La duplicación era en la vista al pedir los datos del socio y del profesor, que los considera iguales. Para solucionarlo creamos una función para pedir los datos genéricos de una persona.

5.6 Mejoras realizadas

Cuando vimos las métricas de nuestro proyecto. Nos tuvimos que plantear una serie de cambios para mejorar la calidad de nuestro programa. En algunos casos fueron verdaderamente duros, ya que tuvimos que cambiar la estructura de una parte de él de forma radical. No obstante, estos cambios se vieron generosamente recompensados al final. A continuación detallaremos uno a uno los cambios hechos sobre el programa.

- Corrección de la complejidad ciclomática, cumplimiento del estándar PEP8 y documentar el código
- Creación de patrones DAO para cada entidad en lugar de una única clase monolítica
- Mejorar índice de mantenibilidad
- Reestructuración del proyecto en subpaquetes de python

5.6.1 Corrección de la complejidad ciclomática

Para esta corrección todo lo que podíamos hacer era reestructurar un poco los bucles existentes en nuestra aplicación. Sobre todos aquellos correspondientes a la manipulación de los ficheros.

Además, se corrigieron todos los errores que incumplían el PEP8 y se documentó el código.

5.6.2 Creación de patrones DAO. Mejorar MI.

Como nuestra aplicación tenía concentrado todo el manejo de datos en una única clase, la única forma de mejorar el MI era separar dicha clase en distintas subclases, tratando cada subclase los datos de una entidad.

Esta reestructuración del proyecto fue la que trajo más dificultades. Sin embargo, también mejoró mucho la calidad general del proyecto.

5.6.3 Reestructuración del proyecto en subpaquetes

A partir de este punto teníamos que empezar a realizar los test unitarios de nuestras clases y generar la documentación, por ese motivo decidimos reestructurar el proyecto en tres directorios: src, tests y doc. Estando el código de la aplicación en src, el código de los tests en test y en doc la parte referida a la documentación.

6 Tests unitarios

A la hora de realizar los tests unitarios nos dimos cuenta de que nuestro diseño no era el más apropiado para testearlo de esta forma; el MVC tiene muchos elementos de alto nivel que no pueden ser testeados con un simple test unitario.

Es por ello que hay clases que tienen un bajo o nulo coverage. La clase "Main" tiene un coverage nulo porque es la vista en terminal de nuestra aplicación. Evidentemente, nos era imposible crear tests unitarios de la vista. Es por ello por lo que decidimos crear tests unitarios de las funciones que validaban todo el input en la vista y que por lo tanto, nos ayudaría a determinar en cierta forma la seguridad y fiabilidad de nuestro sistema ante un input incorrecto.

6.1 Diseño de los tests

Al ser nuestro programa un MVC controlador nuestros tests se han enfocado en un primer lugar en verificar que toda la información que recoge la vista es correcta para nuestro programa y no generará errores. Una vez verificados los inputs, pasamos a testear de forma unitaria cada uno de los modelos de nuestra aplicación. Con estos test verificamos que todas las acciones unitarias de recogida y almacenado de datos se realizan correctamente.

Con esto ya cubrimos todo lo posible con tests unitarios. Las clases individuales, como Socio, Reserva, etc no tiene test unitarios propios porque solo dispone de constructores, getters, setters y to_string, algo que no necesita ser testeado.

Además, como disponemos de un MVC, la parte crítica de nuestra aplicación sería la integración del modelo y el controlador principalmente. Para ello, hemos realizado algunos test de integración, que verifican que controlado manda las órdenes adecuadas al modelo apropiado y este las realiza.

6.2 Tests sobre club

Como el Club es nuestro controlador, en este apartado los únicos tests unitarios que se pueden realizar son para validar información. En este apartado nos centraremos en aquellos tests que validan la información recogida por el Main (vista del programa).

6.2.1 Test "validar_dni"

Esta función se ocupa de verificar que el DNI insertado cumple con el formato establecido. Para ponerla a prueba se la sometió a los siguientes casos de input:

- DNI correcto.
- DNI correcto con el carácter minúsculo.
- Se inserta una cadena cualquiera.
- Se inserta una cadena de 9 dígitos.
- Se inserta una cadena vacía.
- Se Inserta un DNI con formato correcto con un '-' como separador entre los dígitos y el carácter.
- Se inserta un DNI de 8 números y un carácter insertados en orden incorrecto.

Obsérvese que por simplicidad a la hora de probar la aplicación no se ha verificado que los DNIs sean auténticos, solo se verifica que consten de 8 dígitos y una letra al final. Esta funcionalidad, no obstante, se podría implementar fácilmente en el momento de poner este software en producción.

6.2.2 Test "validar_instalacion"

Esta función se ocupa de verificar que la instalación o el material seleccionado se encuentran en la base de datos del club. Para ello se dispuso los siguientes casos:

- Insertar una instalación existente.
- Insertar un extra existente.
- Insertar un material existente.
- Insertar un código identificador de material no existente.
- Insertar una instalación inexistente.

6.2.3 Test "validar_email"

Se ocupa de verificar que un email sigue la estructura correcta: name@server.com. Para ello se dispuso de los siguientes casos:

- Email correcto
- Cadena que no respeta el formato de email

6.2.4 Test "validar_telefono"

Esta función verifica que el teléfono insertado es puramente numérico y que además cumple con la longitud apropiada, 9 dígitos. Para probarla se dispuso de los siguientes casos:

- Teléfono correcto
- Teléfono numérico incompleto
- Cadena alfanumérica

6.2.5 Test "validar_fecha":

Esta función verifica que la fecha insertada cumple con el formato de nuestra aplicación DD/MM/YY HH. Para testearla se dispuso de los siguientes casos:

- Fecha con el formato correcto
- Fecha usando – como separadores
- Fecha con un día erróneo
- Fecha con mes erróneo
- Fecha con hora errónea

Una vez que tuvimos estos tests básicos para asegurar el buen funcionamiento de nuestro programa. Procedimos a testear cada uno de los elementos de bajo nivel de nuestro programa; las clases DAO.

6.3 Tests sobre "conexion_socio"

En este apartado se realizan los test unitarios del modelo de socio. Estos test verifican cada una cada una de las operaciones CRUD que se realiza sobre la clase Socio.

6.3.1 Test "sacar_socio"

Esta función se ocupaba de devolver un socio que esté guardado en el sistema para realizar alguna operación con él o simplemente para acceder a sus datos. Para ello el test realiza los siguientes pasos:

1. Se crea un socio ficticio.
2. Se guarda en el fichero.
3. Se comprueba que se pode encontrar cargado en el programa y que se pueden obtener sus datos.
4. Se borra ese record.

6.3.2 Test "guardar_socio"

Esta función se ocupa de guardar un socio en la base de datos del sistema. Para ello el test realiza los siguientes pasos:

1. Se crea un socio ficticio.
2. Se guarda en el fichero.
3. Se comprueba está guardado correctamente en el fichero.
4. Se borra ese record.

6.3.3 Test "dar_baja_socio"

Esta función se ocupa de dar de baja un socio en la base de datos, es decir, poner su estado a False. Para testearla se realizan los siguientes pasos:

1. Se ha un socio ficticio.
2. Se guardó en el fichero.
3. Se dio de baja a ese socio.
4. Se comprobó que se había guardado correctamente en el fichero.
5. Se borró ese record.

6.3.4 Test "cambiar_socio"

Esta función edita los datos de un socio. Para testearla se realiza lo siguiente:

1. Se guarda un socio ficticio en el fichero
2. Se editan los datos del socio ficticio sus datos
3. Se comprueban que se han guardado los datos correctamente en el fichero
4. Se borra el socio ficticio.

6.4 Tests sobre "conexion_profesor"

En este apartado se realizan los test unitarios del modelo de profesor. Estos test verifican cada una cada una de las operaciones CRUD que se realiza sobre la clase profesor.

6.4.1 Test "sacar_profesor"

Esta función se ocupaba de devolver un profesor guardado en el sistema para obtener sus datos. Los pasos a realizar en el test son:

1. Se crea un profesor ficticio.
2. Se guarda en el fichero.
3. Se obtiene el profesor del fichero y se verifican sus datos.
4. Se borra ese record.

6.4.2 Test "guardar_profesor":

Esta función se ocupa de guardar un profesor en la base de datos del sistema. Para ello hacen los

siguientes pasos:

1. Se crea un profesor ficticio.
2. Se guarda en el fichero.
3. Se comprueba que está guardado correctamente.
4. Se borra ese record.

6.4.3 Test "dar_baja_profesor":

Esta función se ocupa de dar de baja un profesor en la base de datos. Para testearla estos son los pasos:

1. Se crea un profesor ficticio.
2. Se guarda en el fichero.
3. Se da de baja a ese profesor.
4. Se comprueba que los datos se han guardado correctamente.
5. Se borra ese record.

6.4.4 Test "cambiar_profesor":

Esta función edita los datos de un profesor. Para testearla se hace:

1. Se crea un profesor ficticio.
2. Se guarda en el fichero.
3. Se editan sus datos.
4. Se comprueba que se han guardado correctamente en el fichero
5. Se borra ese record.

6.5 Tests sobre "conexion_alquiler"

En este apartado se realizan los test unitarios del modelo de alquiler. De este modelo solo se ha podido testear la operación de guardar el alquiler en el fichero. Esto es debido a que alquiler depende del modelo de reserva, y el resto de test de este modelo ya no sería de carácter unitario.

6.5.1 Test "guardar_alquiler_fichero":

Esta función se ocupaba de guardar en la base de datos un alquiler en el fichero. Para su testeo se realizan los siguientes pasos:

1. Se crea un mock de instalación y socio para crear una reserva ficticia.
2. Se crea una reserva con esos mocks
3. Se crea un alquiler con la reserva creada.
4. Se guarda el alquiler ficticio en un fichero ficticio también.
5. Se comprueba que los datos están en el fichero.
6. Se elimina el fichero.

6.6 Tests sobre "conexion_instalacion"

En este apartado se realizan los test unitarios del modelo de instalacion. Como sobre instalación no realizamos todas las operaciones CRUD, solo se testea el obtener una instalación desde el fichero.

6.6.1 Test "sacar_instalacion":

Esta función se encarga de devolver una instalación guardada en la base de datos. Para testearla se comprueban los siguientes casos:

- Se intenta sacar una instalación inexistente.
- Se saco una instalación existente.

6.7 Tests sobre "conexion_torneo"

En este apartado se realizan los test unitarios del modelo de torneo. Debido a las relaciones de torneo con otros modelos, solo se pudieron realizar tres test unitarios.

6.7.1 Test "guardar_torneo_fichero":

Esta función se ocupa de guardar en el fichero los datos de un torneo. Para testearla se hizo lo siguiente:

1. Se crea un mock de socio, que será el socio que participa en el torneo (8 veces).
2. Se crea un torneo con ese mock de socio y se guarda en el fichero.
3. Se comprueba que se ha guardado de forma satisfactoria y están sus datos.
4. Se borra el fichero.

6.7.2 Test "borrar_torneo":

Esta función se encarga de borrar un torneo de la base de datos. Para testearla se hace lo siguiente:

1. Se crea un torneo (con mock de socios como en el test anterior).
2. Se guarda en el fichero.
3. Se ejecuta la función de borrar_torneo.
4. Se comprueba que el torneo ficticio no existe en el fichero.

6.7.3 Test "sacar_torneo":

Esta función se encarga de devolver un torneo guardado en la base de datos. Para testearla se comprueban los siguientes casos:

- Se intenta sacar un torneo ya existente (calidad).
- Se intenta obtener un torneo no existente.

6.8 Tests sobre "conexion_reserva"

En este apartado se realizan los test unitarios del modelo de reserva. Aunque reserva dependa del modelo de socio e instalación para funcionar correctamente, gracias a los mock se pudo comprobar todas las funcionalidades de este modelo.

6.8.1 Test "guardar_reserva":

Se ocupa de guardar una reserva en la base de datos. Para testearla se hace lo siguiente:

1. Se crea una reserva ficticia, utilizando un mock de socio y de instalación.
2. Se guarda esa reserva en el fichero.
3. Se comprueba que se guardó correctamente
4. Se borra la reserva creada para no modificar el fichero original.

6.9 Test "borrar_reserva":

Esta función se encarga de borrar una reserva de la base de datos. Para testearla se hacen los siguientes pasos:

1. Se crea una reserva ficticia, utilizando mocks.
2. Se guarda en el fichero.
3. Se borra para testear la función a analizar
4. Se comprueba que no existen los datos de la reserva ficticia.

6.9.1 Test "verificar_reserva":

Esta función se encarga de verificar que la reserva no exista ya en la base de datos. Para testearla prueban los siguientes casos:

- Se verifica una fecha de una reserva de prueba ya existente.
- Se verifica una fecha para una reserva disponible.

6.9.2 Test "sacar_reserva":

Esta función encarga de obtener los datos de una reserva existente. Para testearla prueban los siguientes casos:

- Se obtiene una reserva correcta.
- Se intenta obtener una reserva con un dni correcto y una fecha incorrecta.
- Se intenta obtener una reserva con un dni incorrecto y una fecha correcta.

6.10 Tests de integración

En este apartado realizamos algunos test de integración sobre la clase Club para comprobar que esta se integra bien con los modelos correspondientes. Estos tests actúan sobre distintas entidades, por ese motivo los consideramos de integración, aunque quizás teóricamente no se realicen como deberían, verifican la interactuación del Club con los modelos, (conexion_reserva, conexión_socio, etc).

En estos test no creamos datos ficticios y luego los eliminamos porque lo complicaría demasiado. De forma que algunos de estos test alteran los ficheros reales de forma permanente, pero por supuesto no alteran el correcto funcionamiento de la aplicación.

Los tests de integración realizados verifican el correcto funcionamiento del controlador a la hora de interactuar con los modelos de socio, profesor y reserva exclusivamente. Estos tests comprueban las operaciones CRUD de cada clase, y sus operaciones son muy similares a las ya descritas en cada un de los modelos. Por ese motivo no se explicarán cada uno de forma individual.

Con estos tests, se puede asegurar la calidad de la aplicación tanto a la hora de realizar operaciones atómicas como cuando interactúan unas clases con otras.

6.11 Errores corregidos con los tests

Los tests unitarios aportaron pocos errores a la aplicación, lo cual no es del todo satisfactorio pero una vez revisados de nuevo se comprobó que no había muchos más tests a realizar. Sin embargo, los tests de integración si desvelaron varios errores.

6.11.1 Borrar reserva

El test unitario de borrar reserva mostró que se eliminaban más reservas de las que debía, ya que al encontrar la reserva a borrar, borrar todas las demás a partir de esta en el fichero. El problema fue que el if que comprobaba la reserva a eliminar estaba mal formulado.

Además, el test de integración mostró que cuando se borraba una reserva no se eliminaba el alquiler asociado. Dicha funcionalidad se añadió en el club.

6.11.2 Crear reserva

El test de integración de crear reserva mostró varios errores. El primero era que se podía crear una reserva sin verificar previamente que el dni del socio era de un socio existente en el club. Para solventar ese problema, el club obtiene el socio con el dni dado y si existía se crea la reserva.

Otro problema con las reservas fue que se podía crear una para una fecha ya existente. Para ello se

añadió la función verificar_reserva, que comprobaba si esa fecha con la instalación a reservar estaba disponible.

Estos errores con reserva también existían con clase, por lo tanto se solventaron ambos con este test de integración.

6.11.3 Crear socio

El test de integración mostró que la aplicación permitía crear varios socios con el mismo dni. Se solventó el error verificando si el socio existe antes de crearlo. Este error también se corrigió para profesor.

7 Métricas v 0.2

Con el proyecto ya finalizado procedimos a analizar las métricas de nuevo y vemos los nuevos resultados. Estos datos reflejan una gran mejora en todos los apartados de las métricas.

7.1 Cobertura de los tests

Ahora el coverage que muestra nuestra aplicación es muy notable. De los datos que se muestran a continuación podemos observar que la mayoría de las clases de conexión tiene un coverage del 100%, las que no lo tiene es porque todas sus operaciones no son atómicas y depende de otras clases. Además, al realizar algunos tests de integración el coverage de Club y Conexión es adecuado.

Las clases de las entidades, no están cubiertas al 100% porque los test de estas clases no aportan nada. Además tenemos los métodos __str__ que se utilizan en la vista.

De forma general, el proyecto tiene un 70% de coverage. Pero ese dato no es muy representativo debido a que 300 líneas de las 1500 de código son de la vista, y esta tiene un 0% de coverage porque no se puede testear. Por lo si ignoramos esas líneas, el coverage estaría en torno al 90 %. Teniendo en cuenta que es una aplicación MVC está bastante bien cubierta con los test unitarios y algunos de integración.

```
---------- coverage: platform linux2, python 2.7.8-final-0 ----------
Name                          Stmts   Miss  Cover   Missing
-----------------------------------------------------------------
src/Alquiler                     22      5    77%   64-68
src/Clase                        11      3    73%   24, 32, 40
src/Club                        137     39    72%   33-35, 49-51, 161, 174-183, 192, 202-203, 212-214, 225-227, 238-240, 249-250, 259-264, 272, 300
src/Conexion                     63     11    83%   35, 42, 96, 112, 151, 169, 178, 212, 221, 231, 239
src/Instalacion                  12      2    83%   14, 42
src/Main                        308    308     0%   1-354
src/Profesor                     19      1    95%   39
src/Reserva                      14      1    93%   50
src/Socio                        49      2    96%   38, 110
src/Torneo                       19      4    79%   53-56
src/__init__                      1      0   100%
src/conexion_alquiler            65     20    69%   30-33, 47-48, 60-75
src/conexion_clase               52     27    48%   34-45, 53-60, 70-81
src/conexion_instalacion         26      0   100%
src/conexion_profesor            93      0   100%
src/conexion_reserva             63      0   100%
src/conexion_socio               83      0   100%
src/conexion_torneo              65      5    92%   75-79
tests/__init__                    1      0   100%
tests/test_club                 121     16    87%   112-119, 157-166
tests/test_conexion_alquiler     31      0   100%
tests/test_conexion_instalacion   9      0   100%
tests/test_conexion_profesor     66      0   100%
```

```
tests/test_conexion_reserva      59     0   100%
tests/test_conexion_socio        66     0   100%
tests/test_conexion_torneo       45     0   100%
-------------------------------------------------------------------
TOTAL                          1500   444    70%
```

7.2 Complejidad ciclomática

La complejidad ciclomática del proyecto sigue siendo muy buena. La hemos mejorado reduciendo el número de if y anidando los bucles lo mejor posible. Prácticamente todos los métodos tienen una complejidad de A, exceptuando algunos referentes al tratamiento de ficheros o de la vista, cuya cifra es 6 o 8. Como nota adicional, muchos de nuestros métodos tiene una cc de 1, no solo de A.

Por lo que podemos afirmar que la complejidad ciclomática del proyecto es de A.

```
src/Alquiler.py
    M 60:4 Alquiler.__str__ - A (2)
    C 4:0 Alquiler - A (1)
    M 8:4 Alquiler.is_devuelto - A (1)
    M 17:4 Alquiler.get_reserva - A (1)
    M 25:4 Alquiler.get_instalaciones - A (1)
    M 34:4 Alquiler.aniadir_instalacion - A (1)
    M 42:4 Alquiler.set_devuelto - A (1)
    M 50:4 Alquiler.__init__ - A (1)
src/conexion_torneo.py
    M 99:4 ConexionTorneo.__listar_torneos - B (6)
    M 49:4 ConexionTorneo.sacar_torneo - A (4)
    C 7:0 ConexionTorneo - A (3)
    M 31:4 ConexionTorneo.guardar_torneo_fichero - A (3)
    M 81:4 ConexionTorneo.borrar_torneo - A (3)
    M 67:4 ConexionTorneo.poner_resultado - A (2)
    M 11:4 ConexionTorneo.__init__ - A (1)
    M 20:4 ConexionTorneo.guardar_torneo - A (1)
src/Main.py
    F 44:0 pedir_datos_persona - B (6)
    F 5:0 pedir_reserva - A (4)
    F 26:0 consultar_reserva - A (3)
src/Club.py
    M 283:4 Club.validar_dni - A (5)
    M 120:4 Club.crear_reserva - A (3)
    M 165:4 Club.crear_torneo - A (3)
    M 252:4 Club.crear_alquiler - A (3)
    M 322:4 Club.validar_telefono - A (3)
    M 350:4 Club.validar_fecha - A (3)
    M 23:4 Club.alta_socio - A (2)
    M 37:4 Club.alta_profesor - A (2)
    M 99:4 Club.dar_baja_socio - A (2)
    M 111:4 Club.dar_baja_profesor - A (2)
    M 150:4 Club.cancelar_reserva - A (2)
    M 307:4 Club.validar_email - A (2)
    M 337:4 Club.validar_instalacion - A (2)
    C 13:0 Club - A (1)
    M 17:4 Club.__init__ - A (1)
    M 53:4 Club.editar_socio - A (1)
    M 65:4 Club.editar_profesor - A (1)
    M 79:4 Club.obtener_socio - A (1)
    M 89:4 Club.obtener_profesor - A (1)
    M 138:4 Club.consultar_reserva - A (1)
    M 185:4 Club.consultar_torneo - A (1)
    M 194:4 Club.introducir_resultado - A (1)
    M 205:4 Club.borrar_torneo - A (1)
    M 216:4 Club.cancelar_clase - A (1)
    M 229:4 Club.obtener_clase - A (1)
    M 242:4 Club.registrar_clase - A (1)
    M 266:4 Club.devolver_alquiler - A (1)
    M 274:4 Club.consultar_alquiler - A (1)
src/conexion_alquiler.py
    M 53:4 ConexionAlquiler.devolver_alquiler - A (5)
    M 93:4 ConexionAlquiler.__listar_alquileres - A (5)
    M 35:4 ConexionAlquiler.sacar_alquiler - A (4)
    C 9:0 ConexionAlquiler - A (3)
    M 77:4 ConexionAlquiler.guardar_alquiler_fichero - A (2)
    M 13:4 ConexionAlquiler.__init__ - A (1)
    M 24:4 ConexionAlquiler.guardar_alquiler - A (1)
src/conexion_clase.py
    M 25:4 ConexionClase.sacar_clase - B (6)
    M 62:4 ConexionClase.borrar_clase - B (6)
    C 10:0 ConexionClase - A (3)
    M 83:4 ConexionClase.__listar_clases - A (3)
    M 14:4 ConexionClase.__init__ - A (1)
    M 47:4 ConexionClase.guardar_clase - A (1)
src/Reserva.py
    C 4:0 Reserva - A (1)
    M 8:4 Reserva.get_fecha - A (1)
    M 16:4 Reserva.get_socio - A (1)
    M 24:4 Reserva.get_instalacion - A (1)
    M 32:4 Reserva.__init__ - A (1)
    M 44:4 Reserva.__str__ - A (1)
src/Socio.py
    M 9:4 Socio.__init__ - A (2)
    C 5:0 Socio - A (1)
    M 32:4 Socio.__str__ - A (1)
    M 42:4 Socio.get_dni - A (1)
    M 50:4 Socio.get_nombre - A (1)
    M 58:4 Socio.get_apellidos - A (1)
    M 66:4 Socio.get_movil - A (1)
    M 74:4 Socio.get_correo - A (1)
    M 82:4 Socio.get_fecha_alta - A (1)
    M 90:4 Socio.get_estado - A (1)
    M 98:4 Socio.cambiar_estado - A (1)
    M 104:4 Socio.set_fecha_alta - A (1)
    M 112:4 Socio.set_nombre - A (1)
    M 120:4 Socio.set_apellidos - A (1)
    M 127:4 Socio.set_correo - A (1)
    M 135:4 Socio.set_movil - A (1)
src/conexion_reserva.py
    M 38:4 ConexionReserva.sacar_reserva - A (5)
    M 58:4 ConexionReserva.verificar_reserva - A (5)
    M 77:4 ConexionReserva.borrar_reserva - A (5)
```

C 8:0 ConexionReserva - A (3)
M 99:4 ConexionReserva.__listar_reservas - A (3)
M 12:4 ConexionReserva.__init__ - A (1)
M 23:4 ConexionReserva.guardar_reserva - A (1)
src/conexion_socio.py
M 98:4 ConexionSocio.cambiar_socio - B (8)
M 73:4 ConexionSocio.dar_baja_socio - A (5)
M 55:4 ConexionSocio.sacar_socio - A (4)
C 9:0 ConexionSocio - A (3)
M 19:4 ConexionSocio.__listar_socios - A (3)m
M 13:4 ConexionSocio.__init__ - A (1)
M 36:4 ConexionSocio.guardar_socio - A (1)
src/Clase.py
C 4:0 Clase - A (1)
M 8:4 Clase.__init__ - A (1)
M 18:4 Clase.get_profesor - A (1)
M 26:4 Clase.get_reserva - A (1)
M 34:4 Clase.__str__ - A (1)
src/Profesor.py
M 10:4 Profesor.__init__ - A (2)
C 6:0 Profesor - A (1)
M 30:4 Profesor.set_salario - A (1)
M 38:4 Profesor.__str__ - A (1)
M 44:4 Profesor.set_jornada - A (1)
M 52:4 Profesor.get_salario - A (1)
M 60:4 Profesor.get_jornada - A (1)
src/conexion_instalacion.py
M 17:4 ConexionInstalacion.sacar_instalacion - A (4)
C 7:0 ConexionInstalacion - A (3)
M 35:4 ConexionInstalacion.__listar_instalaciones - A (3)
M 11:4 ConexionInstalacion.__init__ - A (1)
src/Conexion.py
C 12:0 Conexion - A (1)
M 17:4 Conexion.__init__ - A (1)
M 30:4 Conexion.guardar_socio - A (1)
M 37:4 Conexion.guardar_profesor - A (1)
M 44:4 Conexion.sacar_socio - A (1)
M 53:4 Conexion.sacar_profesor - A (1)
M 62:4 Conexion.dar_baja_socio - A (1)
M 70:4 Conexion.dar_baja_profesor - A (1)
M 78:4 Conexion.sacar_instalacion - A (1)
M 87:4 Conexion.sacar_clase - A (1)
M 98:4 Conexion.guardar_reserva - A (1)
M 106:4 Conexion.guardar_clase - A (1)
M 114:4 Conexion.sacar_reserva - A (1)
M 124:4 Conexion.verificar_reserva - A (1)
M 135:4 Conexion.borrar_reserva - A (1)
M 145:4 Conexion.guardar_alquiler - A (1)
M 153:4 Conexion.sacar_alquiler - A (1)
M 162:4 Conexion.devolver_alquiler - A (1)
M 171:4 Conexion.borrar_clase - A (1)
M 180:4 Conexion.cambiar_socio - A (1)
M 192:4 Conexion.cambiar_profesor - A (1)
M 206:4 Conexion.guardar_torneo - A (1)
M 214:4 Conexion.sacar_torneo - A (1)
M 223:4 Conexion.poner_resultado - A (1)
M 233:4 Conexion.borrar_torneo - A (1)
src/Instalacion.py
C 4:0 Instalacion - A (1)
M 8:4 Instalacion.get_descripcion - A (1)
M 16:4 Instalacion.get_instalacion_id - A (1)
M 24:4 Instalacion.__init__ - A (1)
M 36:4 Instalacion.__str__ - A (1)
src/conexion_profesor.py
M 59:4 ConexionProfesor.dar_baja_profesor - B (8)
M 89:4 ConexionProfesor.cambiar_profesor - B (8)
C 10:0 ConexionProfesor - A (4)
M 41:4 ConexionProfesor.sacar_profesor - A (4)
M 137:4 ConexionProfesor.__listar_profesores - A (3)
M 14:4 ConexionProfesor.__init__ - A (1)
M 20:4 ConexionProfesor.guardar_profesor - A (1)
src/Torneo.py
M 52:4 Torneo.__str__ - A (2)
C 4:0 Torneo - A (1)
M 8:4 Torneo.__init__ - A (1)
M 19:4 Torneo.get_nombre - A (1)
M 27:4 Torneo.get_socios - A (1)
M 35:4 Torneo.get_resultados - A (1)
M 43:4 Torneo.set_resultado - A (1)
tests/test_club.py
C 11:0 TestClub - A (1)
M 15:4 TestClub.crear_fecha_random - A (1)
M 26:4 TestClub.test_validar_dni - A (1)
M 39:4 TestClub.test_validar_instalacion - A (1)
M 50:4 TestClub.test_validar_email - A (1)
M 58:4 TestClub.test_validar_telefono - A (1)
M 67:4 TestClub.test_validar_fecha - A (1)
M 78:4 TestClub.test_obtener_socio - A (1)
M 86:4 TestClub.test_editar_socio - A (1)
M 99:4 TestClub.test_dar_baja_socio - A (1)
M 108:4 TestClub.alta_socio - A (1)
M 121:4 TestClub.test_obtener_profesor - A (1)
M 129:4 TestClub.test_editar_profesor - A (1)
M 144:4 TestClub.test_dar_baja_profesor - A (1)
M 153:4 TestClub.alta_profesor - A (1)
M 168:4 TestClub.test_crear_reserva - A (1)
M 181:4 TestClub.test_consultar_reserva - A (1)
M 193:4 TestClub.test_cancelar_reserva - A (1)
tests/test_conexion_reserva.py
C 14:0 TestConexionReserva - A (1)
M 18:4 TestConexionReserva.test_guardar_reserva - A (1)
M 40:4 TestConexionReserva.test_sacar_reserva - A (1)
M 51:4 TestConexionReserva.test_verificar_reserva - A (1)
M 61:4 TestConexionReserva.test_borrar_reserva - A (1)
tests/test_conexion_torneo.py
C 11:0 TestConexionTorneo - A (2)
M 15:4 TestConexionTorneo.test_guardar_torneo_fichero - A (2)
M 44:4 TestConexionTorneo.test_borrar_torneo - A (2)
M 36:4 TestConexionTorneo.test_sacar_torneo - A (1)
tests/test_conexion_instalacion.py
C 8:0 TestConexionInstalacion - A (2)
M 12:4 TestConexionInstalacion.test_sacar_instalacion - A (1)
tests/test_conexion_alquiler.py
C 15:0 TestConexionAlquiler - A (2)
M 19:4 TestConexionAlquiler.test_guardar_alquiler_fichero - A (1)
tests/test_conexion_socio.py
C 10:0 TestConexionSocio - A (1)
M 14:4 TestConexionSocio.test_guardar_socio - A (1)
M 33:4 TestConexionSocio.test_sacar_socio - A (1)
M 52:4 TestConexionSocio.test_dar_baja_socio - A (1)
M 72:4 TestConexionSocio.test_cambiar_socio - A (1)
tests/test_conexion_profesor.py

```
    C 9:0 TestConexionProfesor - A (1)
    M 13:4 TestConexionProfesor.test_guardar_profesor - A (1)
    M 32:4 TestConexionProfesor.test_sacar_profesor - A (1)
    M 51:4 TestConexionProfesor.test_dar_baja_profesor - A (1)
    M 71:4 TestConexionProfesor.test_cambiar_profesor - A (1)
```

7.3 Índice de mantenibilidad

El índice de mantenibilidad (MI) ha mejorado mucho con la solución de añadir los patrones DAO y redistribuir todos los modelos en diversas clases. También mejoramos el MI lo suficiente para que en el Main sea A también. El añadir la documentación también supuso una mejora de esta métrica.

```
/ src/Alquiler.py - A (87.09)
src/conexion_torneo.py - A (67.68)
src/Main.py - A (22.52)
src/Club.py - A (60.31)
src/conexion_alquiler.py - A (66.71)
src/conexion_clase.py - A (69.64)
src/Reserva.py - A (88.04)
src/Socio.py - A (72.77)
src/__init__.py - A (100.00)
src/conexion_reserva.py - A (66.79)
src/conexion_socio.py - A (61.82)
src/Clase.py - A (92.71)
src/Profesor.py - A (79.88)
src/conexion_instalacion.py - A (81.74)
src/Conexion.py - A (100.00)
src/Instalacion.py - A (94.00)
src/conexion_profesor.py - A (59.16)
src/Torneo.py - A (85.30)
tests/test_club.py - A (74.95)
tests/test_conexion_reserva.py - A (72.46)
tests/test_conexion_torneo.py - A (81.31)
tests/test_conexion_instalacion.py - A (100.00)
tests/test_conexion_alquiler.py - A (100.00)
tests/__init__.py - A (100.00)
tests/test_conexion_socio.py - A (65.50)
tests/test_conexion_profesor.py - A (65.40)
```

7.4 Métricas crudas

Al añadir la documentación las métricas crudas han mejorado en el apartado (C + M %L) únicamente. El resto de métricas no han mejorado mucho en sí porque el proyecto no tiene otro tipo de comentarios, con la documentación generada con sphinx ya queda suficientemente detallado la aplicación. A continuación se muestran todas las métricas crudas del proyecto.

```
src/Alquiler.py
    LOC: 68
    LLOC: 30
    SLOC: 60
    Comments: 0
    Multi: 38
    Blank: 8
    - Comment Stats
        (C % L): 0%
        (C % S): 0%
        (C + M % L): 56%
src/conexion_torneo.py
    LOC: 119
    LLOC: 75
    SLOC: 110
    Comments: 0
    Multi: 43
    Blank: 9
    - Comment Stats
        (C % L): 0%
        (C % S): 0%
        (C + M % L): 36%
src/Main.py
    LOC: 354
    LLOC: 323
    SLOC: 336
    Comments: 0
    Multi: 16
    Blank: 18
    - Comment Stats
        (C % L): 0%
        (C % S): 0%
```

```
      (C + M % L): 5%
src/Club.py
   LOC: 365
   LLOC: 174
   SLOC: 337
   Comments: 0
   Multi: 192
   Blank: 28
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 53%
src/conexion_alquiler.py
   LOC: 116
   LLOC: 76
   SLOC: 107
   Comments: 0
   Multi: 38
   Blank: 9
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 33%
src/conexion_clase.py
   LOC: 98
   LLOC: 60
   SLOC: 90
   Comments: 0
   Multi: 34
   Blank: 8
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 35%
src/Reserva.py
   LOC: 51
   LLOC: 21
   SLOC: 45
   Comments: 0
   Multi: 30
   Blank: 6
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 59%
src/Socio.py
   LOC: 149
   LLOC: 68
   SLOC: 132
   Comments: 0
   Multi: 80
   Blank: 17
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 54%
src/__init__.py
   LOC: 1
   LLOC: 1
   SLOC: 1
   Comments: 0
   Multi: 0
   Blank: 0
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 0%
src/conexion_reserva.py
   LOC: 116
   LLOC: 72
   SLOC: 108
   Comments: 0
   Multi: 41
   Blank: 8
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 35%
src/conexion_socio.py
   LOC: 138
   LLOC: 95
   SLOC: 125
   Comments: 0
   Multi: 37
   Blank: 13
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 27%
src/Clase.py
   LOC: 40
   LLOC: 16
   SLOC: 35
   Comments: 0
   Multi: 24
   Blank: 5
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 60%
src/Profesor.py
   LOC: 66
   LLOC: 29
   SLOC: 58
   Comments: 0
   Multi: 35
   Blank: 8
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 53%
src/conexion_instalacion.py
   LOC: 50
   LLOC: 32
   SLOC: 46
   Comments: 0
   Multi: 18
   Blank: 4
   - Comment Stats
      (C % L): 0%
      (C % S): 0%
      (C + M % L): 36%
src/Conexion.py
   LOC: 239
   LLOC: 88
   SLOC: 212
```

Comments: 0
Multi: 149
Blank: 27
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 62%
src/Instalacion.py
LOC: 42
LLOC: 17
SLOC: 37
Comments: 0
Multi: 25
Blank: 5
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 60%
src/conexion_profesor.py
LOC: 151
LLOC: 106
SLOC: 138
Comments: 0
Multi: 39

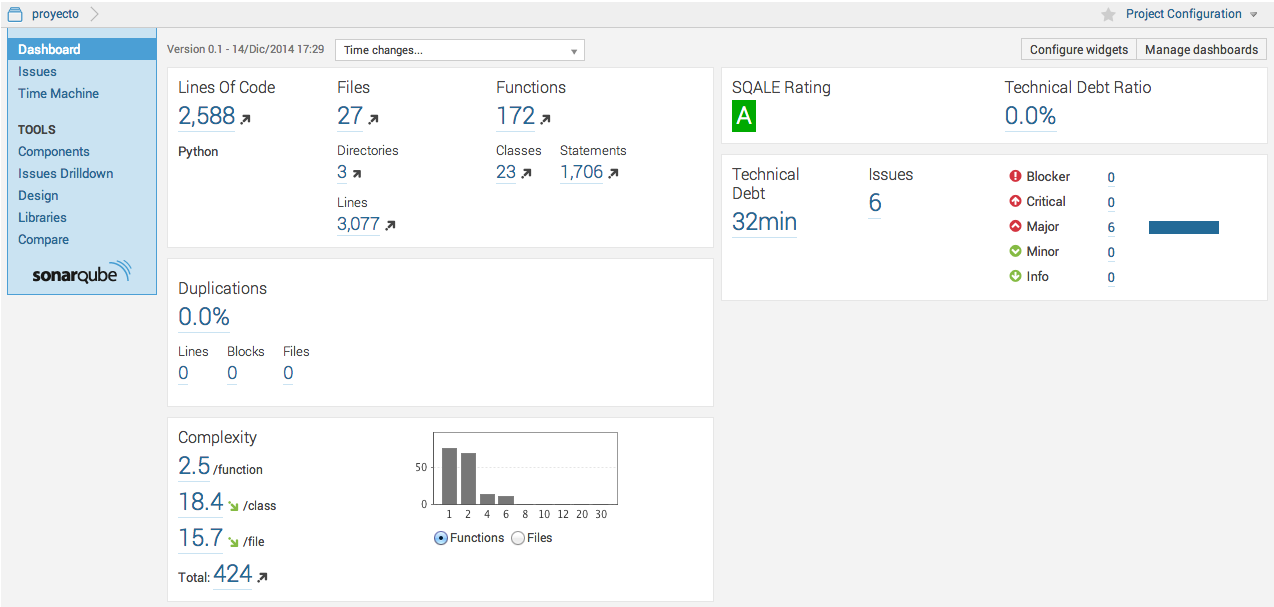
tests/test_club.py
LOC: 203
LLOC: 139
SLOC: 185
Comments: 0
Multi: 56
Blank: 18
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 28%
tests/test_conexion_reserva.py
LOC: 79
LLOC: 66
SLOC: 74
Comments: 0
Multi: 15
Blank: 5
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 19%
tests/test_conexion_torneo.py
LOC: 63
LLOC: 50
SLOC: 59
Comments: 0
Multi: 12
Blank: 4
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 19%
tests/test_conexion_instalacion.py
LOC: 18
LLOC: 11
SLOC: 15
Comments: 0
Blank: 13
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 26%
src/Torneo.py
LOC: 56
LLOC: 26
SLOC: 49
Comments: 0
Multi: 30
Blank: 7
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 54%
**** Total ****
LOC: 2219
LLOC: 1309
SLOC: 2026
Comments: 0
Multi: 869
Blank: 193

Multi: 6
Blank: 3
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 33%
tests/test_conexion_alquiler.py
LOC: 39
LLOC: 33
SLOC: 37
Comments: 0
Multi: 6
Blank: 2
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 15%
tests/__init__.py
LOC: 1
LLOC: 1
SLOC: 1
Comments: 0
Multi: 0
Blank: 0
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 0%
tests/test_conexion_socio.py
LOC: 90
LLOC: 75
SLOC: 84
Comments: 0
Multi: 15
Blank: 6
- Comment Stats
(C % L): 0%
(C % S): 0%

(C + M % L): 17%
tests/test_conexion_profesor.py
LOC: 90
LLOC: 75
SLOC: 85
Comments: 0
Multi: 15
Blank: 5
- Comment Stats
(C % L): 0%
(C % S): 0%
(C + M % L): 17%
** Total **
LOC: 583
LLOC: 450
SLOC: 540
Comments: 0
Multi: 125
Blank: 43

7.5 Duplicidad de código

Volvemos a analizar el proyecto con sonarqube. Los resultados obtenidos son muy satisfactorios. Se ha eliminado la duplicidad de código y además la deuda técnica muestra un 0%. Los errores que aparecen son algunos relacionados con la vista, pero no alteran nada los buenos resultados. A su vez, modificamos algunas reglas de sonarquebe para que se adaptarán a nuestro proyecto, como por ejemplo el uso de la sentencia print.



8 Conclusiones

Gracias a este proyecto nos hemos dado cuenta de lo importante que es tener en cuenta la calidad de una aplicación desde el principio. Nosotros cometimos el error de centrarnos demasiado en terminar la funcionalidad y ello nos condujo a una gran reestructuración de nuestro código. Estos cambios detectados tan tarde nos han hecho invertir mucho tiempo en corregirlos.

Este proyecto también nos ha servido para ver el potencial de un sistema de control de versiones, que facilita mucho el trabajo. Además que hay que tener ciertas normas claras para no corromper los repositorios, ya que en alguna ocasión hemos eliminado trabajo de nuestro compañero. Pero aún así, gracias al github fue fácil recuperar el código.