

EVALUACIÓN	Obligatorio	GRUPO	Todos	FECHA	03/Set/2024
MATERIA	Diseño de Aplicaciones 2				
CARRERA					
CONDICIONES	<p>- Puntaje máximo: 20 puntos</p> <p>- Puntaje mínimo: 0 puntos</p> <p>- Fecha de entrega: 08/Oct/2024 hasta las 21:00 horas en gestion.ort.edu.uy (max. 40Mb en formato zip, rar o pdf) y en GitHub.</p> <p>Uso de material de apoyo y/o consulta</p> <p><u>Inteligencia Artificial Generativa</u></p> <ul style="list-style-type: none"> - Seguir las pautas de los docentes: Se deben seguir las instrucciones específicas de los docentes sobre cómo utilizar la IA en cada curso. - Citar correctamente las fuentes y usos de IA: Siempre que se utilice una herramienta de IA para generar contenido, se debe citar adecuadamente la fuente y la forma en que se utilizó. - Verificar el contenido generado por la IA: No todo el contenido generado por la IA es correcto o preciso. Es esencial que los estudiantes verifiquen la información antes de usarla. - Ser responsables con el uso de la IA: Conocer los riesgos y desafíos, como la creación de “alucinaciones”, los peligros para la privacidad, las cuestiones de propiedad intelectual, los sesgos inherentes y la producción de contenido falso - En caso de existir dudas sobre la autoría, plagio o uso no atribuido de IAG, el docente tendrá la opción de convocar al equipo de obligatorio a una defensa específica e individual sobre el tema <p>IMPORTANTE:</p> <ol style="list-style-type: none"> 1. Inscribirse en la evaluación a través de gestion.ort.edu.uy. 2. Formar grupos de tres personas (no es necesario que pertenezcan al mismo dictado). 3. Subir el trabajo a Gestión (documentación) y a GitHub antes de la hora indicada (ver la sección “RECORDATORIO” al final del documento). 4. Dentro del repositorio privado en la ORG de DA2 de GitHub realizar un <i>merge</i> como <i>release</i> a <i>master</i> (GitFlow) antes de la hora indicada. <p>Aquellos de ustedes que <u>presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos</u>, por favor contactarse con el Coordinador de cursos o Coordinación adjunta antes de las 20:00h del día de la entrega, a través de los mails crosa@ort.edu.uy / posada_l@ort.edu.uy (matutino) / larrosa@ort.edu.uy (nocturno), o vía Ms Teams.</p>				

Obligatorio 1 de Diseño de Aplicaciones II

Se desea diseñar y desarrollar una aplicación que permita centralizar y gestionar diferentes dispositivos inteligentes en un hogar, creando un entorno integrado y automatizado.

Un dispositivo inteligente es un aparato o sistema electrónico que se conecta a internet y puede ser controlado de forma remota mediante aplicaciones móviles, comandos de voz o integraciones con otros sistemas. Ejemplos comunes incluyen termostatos, cámaras de seguridad, bombillas, enchufes, cerraduras, y altavoces inteligentes.

La plataforma que se espera que se desarrolle debe soportar múltiples usuarios con diferentes roles, como administradores, dueños de empresa, y dueños de hogares. Cada rol tiene diferentes niveles de acceso y permisos dentro de la plataforma, lo que permite una gestión más segura y organizada de los dispositivos.

Además de la gestión de usuarios, se debe abordar la problemática de cómo centralizar la gestión de múltiples dispositivos, que pueden pertenecer a diferentes fabricantes, en una única plataforma.

A continuación, se describen los requerimientos funcionales y no funcionales de la plataforma para los diferentes roles que utilizan la aplicación.

Requerimientos funcionales

Un **administrador** puede realizar lo siguiente:

- **Mantenimiento de cuentas de administrador (*):**
Un administrador tiene la capacidad de crear o borrar cuentas con privilegios de administrador, permitiendo que otros administradores colaboren en el mantenimiento de la aplicación. Cada cuenta de administrador debe contener la siguiente información: nombre, apellido, correo electrónico y contraseña.
- **Creación de cuentas para dueños de empresas de dispositivos inteligentes:**
Un administrador puede crear cuentas para dueños de empresas que fabrican o distribuyen dispositivos inteligentes, permitiendo que se unan a la plataforma y gestionen sus dispositivos compatibles. Cada cuenta de dueño de empresa incluye la siguiente información: nombre, apellido, correo electrónico y contraseña. Al concluir la creación de estas cuentas, se marcan como incompletas hasta que se asocie una empresa a la cuenta.
- **Listar todas las cuentas:**
Un administrador puede visualizar una lista de todas las cuentas registradas en la plataforma. La información para mostrar incluye: nombre, apellido, nombre completo (hoy “nombre + apellido”), rol y fecha de creación de la cuenta.
La lista debe ser accesible mediante **paginación** y permitir **filtrar** por rol y nombre completo (**del lado del servidor**).
- **Listar todas las empresas:**

Un administrador puede visualizar una lista de todas las empresas registradas en la plataforma. La información para mostrar incluye: nombre de la empresa, nombre completo del dueño, correo electrónico del dueño y RUT.

La lista debe ser accesible mediante **paginación** y permitir **filtrar** por nombre de la empresa y nombre completo del dueño (**del lado del servidor**).

Manejo de notificaciones:

Las acciones de los dispositivos que requieren notificar a los usuarios deben generar una notificación para los miembros configurados del hogar. Estas notificaciones generadas, pueden ser consumidas en cualquier momento pudiendo ver el evento ocurrido, dispositivo que ejecuto la acción, si fueron leídas o no, fecha y hora.

Estas pueden ser filtradas por tipo de dispositivo, fecha de creación y si fueron leídas o no.

Un **dueño de empresa** puede realizar lo siguiente:

- **Creación de una empresa (*):**

Un dueño de empresa, cuya cuenta esté marcada como incompleta, puede crear y asociar una única empresa a su cuenta. La información requerida para registrar la empresa incluye: nombre, logotipo (la ruta absoluta a la imagen o una URL) y RUT.

- **Registrar cámaras de seguridad:**

Los dueños de empresas con cuentas completas pueden registrar diversos modelos de cámaras inteligentes en la plataforma. La información necesaria para registrar una cámara incluye: nombre, número de modelo, si es para uso interior y/o exterior, descripción, fotografías, y compatibilidad o no con las siguientes funciones: detección de movimiento y detección de persona.

- o **Acciones soportadas:**

- **Detección de movimiento (*):**

- Una cámara, cuando está en línea, puede detectar movimiento y generar una notificación para los miembros del hogar configurados.

- **Detección de persona:**

- Una cámara, cuando está en línea, puede identificar la persona que visualiza y generar una notificación para los miembros del hogar configurados.

- **Registrar sensores de ventanas:**

Los dueños de empresas con cuentas completas pueden registrar diferentes modelos de sensores inteligentes en la plataforma. La información requerida para registrar un sensor incluye: nombre, número de modelo, descripción, fotografías.

- o **Acciones soportadas:**

- **Estado de apertura/cierre:**

- Un sensor, cuando está en línea, puede indicar se abre o cierra y generar una notificación para los miembros del hogar configurados.

Una **persona no autenticada** puede realizar lo siguiente:

- **Autenticación mediante credenciales:**

La plataforma permite a los usuarios autenticarse utilizando credenciales válidas para acceder a su cuenta. Las credenciales requeridas para la autenticación son el correo electrónico y la contraseña.

- **Creación de cuenta para dueño de hogar:**

Una persona puede crear una cuenta en la plataforma como dueño de hogar, lo que le permitirá acceder y gestionar sus hogares con dispositivos inteligentes. La información requerida para crear la cuenta incluye: nombre, apellido, foto de perfil (ruta absoluta a la imagen o una URL), correo electrónico y contraseña.

Una **persona autenticada** puede realizar lo siguiente:

- **Listado de dispositivos:**

Una persona autenticada puede visualizar una lista de los dispositivos registrados en la plataforma. La información mostrada para cada dispositivo incluye: nombre, modelo, foto principal y nombre de la empresa propietaria del dispositivo.

La lista debe ser accesible mediante **paginación** y permitir **filtrar** por nombre del dispositivo, modelo, nombre de la empresa y tipo de dispositivo (**del lado del servidor**).

- **Listado de tipos de dispositivos soportados:**

Una persona autenticada puede consultar una lista de los tipos de dispositivos que la plataforma admite, mostrando el nombre de cada tipo de dispositivo. Se sabe que estos tipos no suelen cambiar constantemente.

Un **dueño de hogar** puede realizar lo siguiente:

- **Creación de hogares:**

Una persona autenticada puede crear múltiples hogares en la plataforma, donde será designado como dueño del hogar. La información necesaria para registrar un hogar incluye: dirección (calle principal, número de puerta), ubicación geográfica (latitud y longitud), y el número de miembros que residen en el hogar.

- **Agregar miembros al hogar:**

El dueño del hogar puede agregar cuentas existentes en la plataforma a unirse al hogar como miembros. La cantidad de miembros debe ser menor o igual a la cantidad soportada del hogar. Los miembros de un hogar pueden tener permisos para asociar dispositivos al hogar y/o listar dispositivos del hogar.

- **Asociar dispositivos al hogar (*):**

El dueño del hogar o cualquier miembro con el permiso adecuado puede asociar dispositivos, previamente registrados en la plataforma, a cualquiera de los hogares bajo su control. Esta asociación genera un hardware id y un estado de conexión (conectado / desconectado).

- **Listado de miembros de un hogar:**

El dueño del hogar puede ver una lista completa de los miembros asociados a su hogar, visualizando la siguiente información: nombre completo, correo electrónico, foto de perfil de cada miembro, el listado de permisos otorgados y si pertenece al listado de miembros que deben ser notificados.

- **Listado de dispositivos en un hogar:**

El dueño del hogar o cualquier miembro con el permiso adecuado puede ver todos los dispositivos instalados en su hogar, con información como: nombre, modelo, foto principal, y el estado de conexión del dispositivo.

- **Configuración de notificaciones para miembros:**

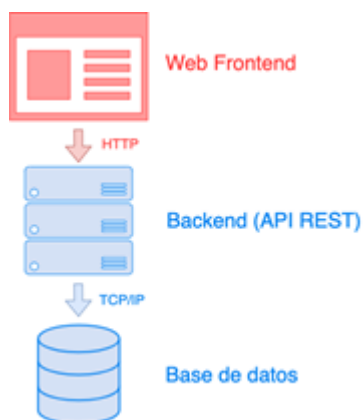
El dueño del hogar puede seleccionar qué miembros del hogar recibirán notificaciones de los dispositivos.

Requerimientos no funcionales:

- **Inicialización con datos predefinidos (seed data):**

Para la primera ejecución de la plataforma, debe existir un conjunto de datos predefinidos (seed data) que permita la creación inicial de administradores y dueños de empresa.

Para resolver el problema se definió la siguiente **arquitectura de alto nivel**:



Artefacto	Descripción
Base de datos	Base de datos relacional (SQL Server) en donde se almacenan los datos de la aplicación.
Api REST + Backend	Toda interacción con el repositorio de datos se realiza mediante un API REST, la que ofrece operaciones para resolver todo lo necesario y el back end donde se implementa la lógica de negocio.
Web Frontend	Aplicación que permite a los usuarios registrarse e interactuar con la aplicación.

Alcance de la primera entrega

Considerando la arquitectura mencionada en la sección anterior, para esta primera **entrega se debe implementar una API REST** que ofrezca todas las operaciones que el alumno considere necesarias para soportar la aplicación descrita en este documento.

Dicho de otra manera, se debe implementar **todo el backend** de la aplicación (lógica de negocio), pero **no el frontend** de la misma (interfaz de usuario). Se espera que, mediante la descripción de la aplicación, y complementando con consultas a aulas cuando considere necesario, el alumno sea capaz de identificar las operaciones que el API debe ofrecer.

Dado que la aplicación no tendrá interfaz de usuario, la forma de interactuar con la misma será mediante un cliente HTTP. Se espera que se utilice **Postman** (<https://www.getpostman.com/>) como cliente, ya que se debe entregar una exportación de una colección de Postman con los end-points definidos para poder probar. Debe tener todas las llamadas a la API preparadas y utilizar la parametrización de Postman en cuanto a la URL a utilizar, token en el header, etc. para no tener que escribir todas las llamadas nuevamente en la demostración.

En caso de no entregar dicha colección, los docentes pueden optar por la no corrección de la funcionalidad. Ver la rúbrica de evaluación en las secciones más abajo.

Implementación

Se debe entregar una implementación del API REST necesaria para soportar la aplicación que se describe. La entrega debe contener una solución de Visual Studio que agrupe todos los proyectos implementados.

La solución debe incluir el código de las pruebas automáticas. Se requiere escribir los casos de prueba automatizados con MSTests, documentando y justificando las pruebas realizadas.

Se espera que la aplicación se entregue con una base de datos con datos de prueba, de manera de poder comenzar las pruebas sin tener que definir una cantidad de datos iniciales. Dichos datos de prueba deben estar adecuadamente especificados en la documentación entregada.

Tecnologías y herramientas de desarrollo

- Microsoft VSCode/ Visual Studio
- Microsoft SQL Server Express 2017
- Postman
- NET Core SDK 8.0/ ASP.NET Core 8.0(C#)
- Entity Framework Core 8.0
- Astah o cualquier otra herramienta UML 2

NOTA: La totalidad y detalle de los requisitos serán relevados a partir de consultas en el foro correspondiente en aulas. Para evitar complejidades innecesarias se realizaron simplificaciones al dominio del problema real.

Independencia de librerías

Se debe diseñar la solución que al modificar el código fuente minimice el impacto del cambio en los componentes físicos de la solución. Debe documentar explícitamente como su solución cumple con este punto. Cada paquete lógico debe ser implementado en un *assembly* independiente, documentando cuáles de los elementos internos al paquete son públicos y cuáles privados, o sea cuáles son las interfaces de cada *assembly*.

Persistencia de los datos

La empresa requiere que todos los datos del sistema sean persistidos en una base de datos. Así, cuando se ejecute la aplicación se empezará con los datos cargados con el último estado guardado antes de cerrar la aplicación.

El diseño debe contemplar el modelado de una solución de persistencia adecuada para el problema utilizando Entity Framework (*Code First*).

Se espera que como parte de la entrega se incluya dos respaldos de la base de datos: uno vacío y otro con datos de prueba. Se debe entregar el archivo *.bak* y también el script *.sql* para ambas bases de datos.

Mantenibilidad

La propia empresa eventualmente hará cambios sobre el sistema, por lo que se requiere un alto grado de mantenibilidad, flexibilidad, calidad, claridad del código y documentación adecuada.

Por lo que el desarrollo de todo el obligatorio debe cumplir:

- Estar en un repositorio **Git**.
- Haber sido escrito utilizando **TDD** (desarrollo guiado por pruebas) lo que involucra otras dos prácticas: escribir las pruebas primero (Test First Development) y Refactoring. De esta forma se utilizan las pruebas unitarias para dirigir el diseño.

Es necesario utilizar **TDD únicamente** para el *back end* y la *API REST*, no para el desarrollo del *front end*.

Se debe utilizar un framework de Mocking (como Moq, <https://www.nuget.org/packages/moq/>) para poder realizar pruebas unitarias sobre la lógica de negocio. En caso de necesitar hacer pruebas para el acceso a datos, deberán de hacerlo con el paquete EF Core Sqlite (<https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.Sqlite>).

- Cumplir los lineamientos de **Clean Code** (capítulos 1 al 10, y el 12), utilizando las técnicas y metodologías ágiles presentadas para crear código limpio.

Control de versiones

La gestión del código del obligatorio debe realizarse utilizando UN ÚNICO repositorio Git de **Github**, apoyándose en el flujo de trabajo recomendado por **GitFlow** (<https://nvie.com/posts/a-successful-git-branching-model/>).

Dicho repositorio debe **pertenecer a la organización de GitHub “IngSoft-DA2”** (<https://github.com/IngSoft-DA2>), en la cual deben estar todos los miembros del equipo.

Al realizar la entrega se debe realizar un *release* en el repositorio y la rama *master* no debe ser afectada luego del hito que corresponde a la entrega del obligatorio.

Evaluación (20 puntos)

Las condiciones de entrega serán evaluadas como si se le estuviese entregando a un cliente real: prolijidad, claridad, profesionalismo, orden, etc.

La entrega debe realizarse antes de las 21:00 horas del día de la entrega, tanto en **gestion.ort.edu.uy** como en el repositorio de GitHub del equipo. A continuación, se detallan los requisitos para cada uno:

1. **Entrega en Gestión (gestion.ort.edu.uy):**

- La documentación completa debe estar en formato PDF e incluir el modelado UML. Es fundamental que todos los diagramas sean claros y legibles.

2. **Entrega en el Repositorio GitHub del equipo:**

- **Carpeta “Código Fuente”:** Contendrá el código fuente de la aplicación, junto con todo lo necesario para compilar y ejecutar la aplicación.
- **Carpeta “Aplicación”:** Incluirá la aplicación compilada en modo release y todos los elementos necesarios para realizar la instalación (deploy) de la misma.
- **Carpeta “Documentación”:** Contendrá la documentación solicitada, incluyendo el modelado UML. Es importante asegurarse de que los diagramas sean claros y legibles en el documento.
- **Carpeta “Base de Datos”:** Incluirá los archivos .bak y .sql. Se debe entregar una versión de la base de datos vacía y otra con datos de prueba.

La documentación para entregar debe dividirse en **4 documentos**:

- Descripción del diseño (PDF que no debe superar las 20 páginas).
- Evidencia del diseño y especificación de la API.
- Evidencia de la aplicación de TDD y Clean Code (PDF que no debe superar las 15 páginas).
- Evidencia de la ejecución de las pruebas de la API con Postman (PDF que no debe superar las 15 páginas).

Todos los documentos deben cumplir con los siguientes elementos del Documento 302 de la facultad (<http://www.ort.edu.uy/fi/pdf/documento302facultaddeingenieria.pdf>):

- Capítulo 3, secciones 3.1 (sin la leyenda), 3.2, 3.5, 3.7, 3.8 y 3.9.
- Capítulo 4 (salvo 4.1).
- Capítulo 5.
- Se **debe incluir en las portadas de cada documento la URL al repositorio del equipo**

Rúbrica de evaluación

Evaluación de	
---------------	--

<p>Descripción del diseño.</p>	<p>El objetivo del documento es demostrar que el equipo pudo diseñar y documentar el diseño de la solución. La documentación debe pensarse para que un tercero (corrector) pueda comprender la estructura y los principales mecanismos del código. O sea, debe servir como guía para entender el código y los aspectos más relevantes del diseño y la implementación.</p> <p>El documento debe organizarse siguiendo el modelo 4+1 haciendo énfasis en las vistas de diseño e implementación. Este documento no debe incluir paquetes o componentes de los proyectos de prueba.</p> <p>Elementos para evaluar:</p> <ul style="list-style-type: none"> ● Descripción general del trabajo (qué hace la solución) y errores conocidos (<i>bugs</i> o funcionalidades no implementadas). ● Diagrama general de paquetes (namespaces) mostrando los paquetes organizados por capas (<i>layers</i>) y sus dependencias. En caso de que haya paquetes anidados, se debe utilizar el conector de <i>nesting</i>, mostrando la jerarquía de dichos paquetes. ● Cada paquete debe tener una breve descripción de responsabilidades y un diagrama de clases. ● Descripción de jerarquías de herencia utilizadas (en caso de que así haya sido) ● Modelo de tablas de la estructura de la base de datos. ● Para aquellas funcionalidades que el equipo entienda como relevantes: <ul style="list-style-type: none"> ○ Diagramas de interacción que muestran las clases involucradas en estas funcionalidades y las interacciones para lograr la funcionalidad. ○ Estas interacciones a mostrar pueden ser del flujo a alto nivel, de un cierto algoritmo específico de su obligatorio, etc ● Justificación del diseño explicando mediante texto y diagramas, haciendo énfasis en: <ul style="list-style-type: none"> ○ La utilización de mecanismos de inyección de dependencias, fábricas, patrones y principios de diseño, etc. Discutir brevemente cómo estos mecanismos apoyan la mantenibilidad de la aplicación. ○ Sus propias decisiones propias de diseño que hacen a su obligatorio. Explicación de estructuras, mecanismos o algoritmos que tienen impacto en la mantenibilidad o desempeño de la solución. 	<p>6</p>
---------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------

	<ul style="list-style-type: none"> ○ Descripción del mecanismo de acceso a datos utilizado. ○ Descripción del manejo de excepciones. ● Diagrama de implementación (componentes) mostrando las dependencias entre los mismos. Justificar el motivo por el cual se dividió la solución en dichos componentes. <p>Se considera como aceptable si:</p> <ul style="list-style-type: none"> ● Los diagramas presentan el uso adecuado de la notación UML. ● La estructura de la solución representa la descomposición lógica (módulos) y de proyectos de la aplicación. ● Las vistas de módulos, de componentes, modelo de datos y los comportamientos documentados sirven como guía para la comprensión del código implementado. ● La taxonomía de paquetes y clases en los diagramas respeta las convenciones de nombre de C# utilizada en la implementación. ● Se justifica y explica el diseño en base al uso de principios y patrones de diseño. Los mismos se implementan correctamente a partir de su objetivo y teniendo en cuenta sus ventajas y desventajas para favorecer o inhibir la calidad de la solución. El objetivo es describir el impacto de las decisiones tomadas para mejorar la mantenibilidad del sistema. No debe limitarse una descripción de lo realizado. ● Correcto manejo de las excepciones e implementación de acceso a base de datos. ● Se describen claramente los errores conocidos. ● La documentación se encuentra bien organizada, es fácil de leer y su formato corresponde con los ítems indicados del documento 302. ● La documentación no debe superar las 20 páginas. 	
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

<p>Evidencia del diseño y especificación de la API.</p>	<p>Documento describiendo la API conteniendo:</p> <ul style="list-style-type: none"> ● Discusión de los criterios seguidos para asegurar que la API cumple con los criterios REST. ● Descripción del mecanismo de autenticación de requests. ● Descripción general de códigos de error (1xx, 2xx, 4xx, 3xx, 5xx). ● Descripción de los <i>resources</i> de la API. <ul style="list-style-type: none"> ○ URL base. ○ Para cada <i>resource</i> describir: <ul style="list-style-type: none"> ▪ Resource. ▪ Description. ▪ Endpoints (Verbo + URI). ▪ Parameters. ▪ Responses (para todos los códigos de estado). ▪ Headers. <p>Se considera como aceptable si:</p> <ul style="list-style-type: none"> ● El diseño de la API REST se basa en buenas prácticas de la industria. Por ejemplo: <ul style="list-style-type: none"> ○ https://developer.spotify.com/documentation/web-api/reference/ ○ https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design ○ Web API Design: The Missing Link by apigee. ● Para generar la documentación se puede utilizar herramientas como swaggerHub (https://swagger.io/tools/swaggerhub/) que permiten generar la especificación de la API en formato electrónico. ● La documentación se encuentra bien organizada, es fácil de leer y su formato corresponde con los ítems indicados del documento 302. 	<p>4.5</p>
----------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------

<p>Evidencia de Clean Code y de la aplicación de TDD</p>	<p>El trabajo se debe desarrollar en su totalidad siguiendo las prácticas de Clean Code y el enfoque de desarrollo TDD. Con el fin de demostrar la correcta ejecución de TDD, para las funcionalidades marcadas con (*), es necesario demostrar la correcta aplicación de esta técnica.</p> <p>El código debe ajustarse a las prácticas recomendadas por Clean Code. Estas apuntan a que el código sea legible por un tercero. La mejor evidencia de la aplicación de Clean Code es que un tercero (los docentes) puedan leer el código con el menor esfuerzo posible.</p> <p>Resultado de la ejecución de las pruebas. Evidencia del código de pruebas automáticas (unitarias y de integración), reporte de la herramienta de cobertura y análisis del resultado. Como parte de la evaluación se va a revisar el nivel de cobertura de los tests sobre el código entregado, por lo que se debe entregar un reporte y un análisis de la cobertura de las pruebas.</p> <p>Ítems para evaluar para cada una de las funcionalidades indicadas:</p> <ul style="list-style-type: none"> ● Descripción de la estrategia de TDD seguida (inside – out o outside - in). ● Informe de cobertura para todas las pruebas desarrolladas. ● Es importante mantener una clara separación de los proyectos de prueba de los de la solución. ● Para las funcionalidades especificadas como prioritarias (*) se debe tener en cuenta: <ul style="list-style-type: none"> ○ Dejar evidencia mediante los commits de que se aplicó TDD mostrando la evolución del ciclo de TDD. ○ Análisis de las métricas de cobertura. Se espera que la métrica para las pruebas del código indicado la cobertura se encuentre en un valor entre 90% y 100% de líneas de código. Se debe realizar un análisis de cualquier desvío de estos valores. <p>Se considera como aceptable si:</p> <ul style="list-style-type: none"> ● El código debe ajustarse a las buenas prácticas de Clean Code. https://www.planetgeek.ch/wp-content/uploads/2013/06/Clean-Code-V2.1.pdf ● Un tercero conocedor de Clean Code pueda leer el código sin dificultad. ● El informe de las pruebas sirve como evidencia de la correcta aplicación de desarrollo guiado por las pruebas (TDD) y 	<p>4.5</p>
-----------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------

	<p>técnicas de refactorio de código. Se justifica claramente el motivo por los cuales se obtuvieron los valores registrados</p> <ul style="list-style-type: none">• La documentación se encuentra bien organizada, es fácil de leer y su formato corresponde con los ítems indicados del documento 302.• La documentación no debe superar las 15 páginas.	
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--


<p>Evidencia de la ejecución de las pruebas de la API con Postman.</p>	<p>Se debe generar colecciones de pruebas en postman para todas las funcionalidades, las cuales el deben incluir en el repositorio.</p> <p>Para las funcionalidades marcadas con (*) se debe documentar los casos de prueba elaborados, incluyendo: valores inválidos, valores límites, ingreso de tipos de datos erróneos, datos vacíos, datos nulos, omisión de campos obligatorios, campos redundantes, Body vacío {}, formato de los mensajes inválidos, pruebas de las reglas del negocio como alta de elementos existentes, baja de elementos inexistentes, etc.</p> <p>Se debe entregar un reporte que muestre evidencia del resultado de ejecutar los casos de prueba especificados para la API con Postman para las funcionalidades marcadas con (*).</p> <p>La evidencia <u>debe</u> registrarse realizando un único video publicado en Youtube, en los cual se pueda apreciar claramente la ejecución de las pruebas. El link a este video debe incluirse en este documento y se debe verificar que se hayan compartido correctamente y que un tercero pueda verlos.</p> <p>Se considera como aceptable si:</p> <ul style="list-style-type: none"> • El documento deja claro que la prueba que se ejecutó y cuál fue el resultado obtenido. • La documentación se encuentra bien organizada, es fácil de leer y su formato corresponde con los ítems indicados del documento 302. • La documentación no debe superar las 15 páginas. 	<p>5</p>
-------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------

RECORDATORIO: IMPORTANTE PARA LA ENTREGA

- **Obligatorios**

La entrega de los obligatorios será en formato digital online, a excepción de algunas materias que se entregarán en Bedelía y en ese caso recibirá información específica en el dictado de la misma.

Los principales aspectos a destacar sobre la **entrega online de obligatorios** son:

1. Ingresá al sistema de Gestión.
2. En el menú, seleccioná el ítem "Evaluaciones" y la instancia de evaluación correspondiente, que figura bajo el título "Inscripto".
3. Para iniciar la entrega hacé clic en el ícono: 
4. Ingresá el número de estudiante de cada uno de los integrantes y hacé clic en "Agregar". El sistema confirmará que los integrantes estén inscriptos al obligatorio y, de ser así, mostrará el nombre y la fotografía de cada uno de ellos. Una vez agregados todos los integrantes, hacé clic en "Crear equipo".

Cualquier integrante podrá:

- **Modificar la integración del equipo.**
- **Subir el archivo de la entrega.**

5. Seleccioná el archivo que deseás entregar. Verificá el nombre del archivo que aparecerá en la pantalla y hacé clic en "Subir" para iniciar la entrega. Cada equipo (hasta 2 estudiantes) debe entregar **un único archivo en formato zip o rar** (los documentos de texto deben ser pdf, y deben ir dentro del zip o rar). El archivo a subir debe tener **un tamaño máximo de 40mb**

Cuando el archivo quede subido, se mostrará el nombre generado por el sistema (1), el tamaño y la fecha en que fue subido.

6. El sistema enviará un e-mail a todos los integrantes del equipo informando los detalles del archivo entregado y confirmando que la entrega fue realizada correctamente.
7. Podés cerrar la pestaña de entrega y continuar utilizando Gestión o salir del sistema.
8. **La hora tope para subir el archivo será las 21:00** del día fijado para la entrega.
9. La entrega se podrá realizar desde cualquier lugar (ej. hogar del estudiante, laboratorios de la Universidad, etc).
10. Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con el Coordinador de cursos o Coordinación adjunta antes de las 20:00h del día de la entrega, a través de los mails crossa@ort.edu.uy / posada_l@ort.edu.uy (matutino) / larrosa@ort.edu.uy (nocturno), o vía Ms Teams.