Obligatorio I de Diseño de Aplicaciones I

Lectura del obligatorio:	01/04/2024
Plazo máximo de entrega:	09/05/2024 hasta las 21 h
Puntaje mínimo / máximo	0/20
Tamaño máximo del equipo	3 estudiantes
No se permiten equipos integrados por estudiantes de diferentes dictados	

DepoQuick es una iniciativa emergente que busca revolucionar el mercado de alquiler de depósitos, proporcionando una solución eficiente y fácil de usar para aquellos que necesitan espacio adicional de almacenamiento. La misión de DepoQuick es simplificar el proceso de alquiler de depósitos, haciendo que sea tan fácil como realizar un pedido en línea.

Para lograr esto, DepoQuick solicita a su equipo desarrollar su Producto Mínimo Viable (MVP). Los requerimientos específicos son los siguientes:

Requerimientos para todos los equipos

Dado que sólo se desea implementar una prueba de concepto, los requerimientos de implementación para el obligatorio son los siguientes:

No funcionales:

1. Debe separar los componentes de la solución de forma que la lógica de negocio y la interfaz de usuario estén independientes.
2. Ambos componentes deben ser desarrollados utilizando C# en .NET Core 6, con Blazor Server App y utilizando GitHub como repositorio de código.
3. Todas las clases deberán ser diseñados y construidos 100% utilizando TDD; y contener la implementación de todas las reglas de negocio que se especifican en la letra. **No incluye la interfaz de usuario.**

Funcionales:

1. **Registro, inicio de Sesión, cierre de sesión:**
    1. Al ingresar a la aplicación por primera vez se llevará al usuario a la página de "Registro" para registrar al "Administrador". Los datos del Administrador son los mismos que para los clientes.
       Unicamente se puede registrar **un solo** administrador.
    2. Registro de clientes:

i. Los clientes podrán ser registrados por el Administrador o podrán auto registrarse. Los datos que se guardarán del cliente son:
    1. Nombre y apellido (texto, admite espacios, máximo 100 caracteres)
    2. Email (se deberá verificar que sea un email con formato válido ej. nombre@dominio)
    3. Contraseña (usar input tipo password)
        a. Mínimo 8 caracteres
            i. Un símbolo: #@$.,%
            ii. Una letra minúscula
            iii. Una letra mayúscula
            iv. Un dígito
    4. Verificación de contraseña
        a. Chequear que se ingrese nuevamente la misma contraseña que en 3.

3. Inicio de sesión
   Tanto clientes como administradores podrán ingresar a la aplicación con su email y contraseña.

2. Administración de Promociones (Alta, Baja, Modificación):

El administrador podrá dar de alta Promociones, de las promociones se conoce:

- Etiqueta (texto alfanumérico, admite espacios, máximo 20 caracteres)
- Porcentaje de descuento (mínimo 5% máximo 75%)
- Validez (rango de fechas a día completo, desde – hasta)

3. Alta de depósito

El alta de depósitos solamente podrá ser efectuada por el "Administrador**".** Se guarda:

- Área: A, B, C, D ó E.
- Tamaño: Pequeño, mediano ó grande
- Climatización: Si / No
- Promociones
    - Si tiene, se deberá poder seleccionar una o múltiples promociones (ver punto 2).

4. Baja de depósito

El baja de depósitos solamente podrá ser efectuada por el "Administrador**".** Solamente se podrán dar de baja depósitos que no estén reservados.

5. **Cálculo de precio de depósito**

Se debe generar un algoritmo que en función de las siguientes variables calcule el precio de alquiler de un depósito:

Tamaño del Depósito: El precio base varía según el tamaño (pequeño, mediano, grande).
- Pequeño: $50 por día
- Mediano: $75 por día
- Grande: $100 por día

Duración del Alquiler: Descuentos aplicados según la duración.
- Menos de 7 días: Sin descuento
- Entre 7 y 14 días: 5% de descuento sobre el precio total
- Más de 14 días: 10% de descuento sobre el precio total

Climatización: Si el depósito requiere climatización, se añade un suplemento.
- Sin climatización: Sin suplemento
- Con climatización: $20 adicionales por día

Oferta especial: Si existe alguna promoción vigente para ese tipo de depósito se aplica la promoción (Ej.: 20% off) sobre el precio con los descuentos aplicados anteriormente.

6. **Reserva de Depósitos:**

Los clientes podrán realizar reservas de los depósitos. Para la reserva se pide que se elija la fecha de comienzo y fin de la reserva. La reserva no implica que el cliente ya tiene el depósito asignado, faltará la confirmación por parte del administrador.

Antes de hacer efectiva la reserva se informará el costo aplicando el algoritmo desarrollado para la parte 5 y luego de ver el precio el cliente efectuará la reserva.

Las reservas no tienen en cuenta disponibilidad del depósito, es decir: puede reservarse el mismo deposito para la misma fecha múltiples veces para múltiples usuarios.

7. **Gestión de Reservas:**

El administrador podrá visualizar las reservas activas de todos los depósitos. Las reservas pueden estar marcadas como: aprobadas o rechazadas. El administrador podrá aceptar o rechazar la reserva; si la rechaza deberá indicar por qué ingresando un texto libre de máximo 300 caracteres.

El cliente podrá visualizar sus reservas y ver si están aceptadas o rechazadas.

Requerimientos adicionales para equipos de tres estudiantes

Para los equipos de tres estudiantes se espera que se resuelvan o modifiquen los siguientes requerimientos:

1. **Valoraciones y Comentarios**

Luego de que se termina el alquiler de un depósito, el cliente podrá hacer una valoración de entre una y cinco estrellas del servicio. Además, podrá agregar un comentario de máximo 500 caracteres.

Esta valoración y sus comentarios deben ser reflejada al usuario en el proceso de reserva.

2. **Registro (log) de acciones**

Se deberá llevar un registro de las acciones por usuario. Este log de acciones se podrá visualizar solamente por el "Administrador". Las acciones que se registrarán son:

- Inicio de sesión
- Cierre de sesión
- Cliente creó valoración

3. **Emisión de estadísticas**

Se podrá visualizar por zonas (indicadas en el punto 3 de los requerimientos generales) la cantidad de dinero que se ha generado en un rango de fechas por los alquileres y por otro lado en un gráfico de barras el número de depósitos alquilados.

Nota sobre requerimientos (todos los equipos)

Nota: **La totalidad y detalle de los requisitos serán relevados a partir de consultas en el foro correspondiente en aulas**, las que deben tener un título descriptivo de su contenido. Tener en cuenta al momento de planificar el trabajo que las respuestas en Aulas **pueden demorar hasta 48 horas**. Para evitar complejidades innecesarias se realizaron simplificaciones al dominio del problema real.

Entrega del obligatorio:

A continuación, se describen los requisitos esperados para la entrega del obligatorio:

Diseño e Implementación:

Se debe entregar una aplicación que contemple toda la funcionalidad descrita en este documento. **Para esta primera instancia la solución guardará toda la información en memoria**, es decir, no es necesario el uso de base de datos (esto se implementará en la segunda entrega).

El desarrollo de todo el obligatorio debe cumplir:

- Estar en un repositorio **Git**, siguiendo **Gitflow**.
- **El nombre del repositorio debe estar conformado de la siguiente forma**:
    - NumeroEstudiante1_NumeroEstudiante2
- Haber sido desarrollado 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.
- Cumplir los lineamientos de **Clean Code** (capítulos 1 al 10, y el 12), utilizando las técnicas y metodologías ágiles presentadas para crear código limpio.
- Utilizar las convenciones de codificación (“idioms”) de C#: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions
- Se requiere escribir los casos de prueba automatizados con el framework de unit tests visto en el curso, **documentando** y **justificando** las pruebas realizadas (ver documentación).
- Como parte de la evaluación se revisará el **nivel de cobertura** de los test sobre el código entregado. Por lo que se debe entregar un reporte y un análisis de la cobertura de las pruebas justificando los valores obtenidos según el diseño elaborado para la solución.

Documentación:

La documentación entregada debe ser en **un solo** documento digital (**límite 15 páginas, sin contar anexos)**, que contenga la siguiente información ordenada e indexada:

1. Carátula de documento como indica el documento 302. **Debe incluirse un link al repositorio de la entrega.**
2. Descripción general del trabajo y del sistema: si alguna funcionalidad no fue implementada o si hay algún error conocido (bug) deben ser descritos aquí.
3. Descripción y justificación de diseño. Para ello se debe incluir:
    a. Diagrama(s) de paquetes mostrando la organización general de la aplicación (**no es necesario incluir el paquete de pruebas**). Se requiere que se justifique el criterio para asignar responsabilidades es los diferentes namespaces. Se puede hacer uso de la siguiente tabla:

Namespace	Clase	Responsabilidad
Dominio	Usuario	....
Dominio.Excepciones	UsuarioException	....

    b. Diagramas de clases: al menos uno por paquete. Los diagramas deberán ser lo más completos y formales posible **respecto a lo que se desea comunicar**. La representación es suficiente con *properties* y métodos, no es necesario atributos.
    c. **Explicación de los mecanismos generales y descripción de las principales decisiones de diseño tomadas**. A modo de ejemplo, se debería describir la forma en que la interfaz de usuario interactúa con el dominio, cómo se almacenan los datos, el manejo de errores y excepciones, o la utilización de polimorfismo.
    d. **Breve análisis de los criterios seguidos para asignar las responsabilidades**.
4. Mantenibilidad y extensibilidad. Es necesario evidenciar una funcionalidad o sección de su aplicación en la cual extender o modificar la funcionalidad es sencillo y tiene el mínimo impacto posible. Se puede complementar con ejemplos.
5. Análisis de dependencias. Es necesario evidenciar para el requerimiento: **Cálculo de precio de depósito** cuales son las clases del dominio que interactúan en dicha funcionalidad y cuáles son las dependencias entre clases analizando la cohesión y acoplamiento.

6. Cobertura de pruebas unitarias con su debido análisis y justificaciones. En caso de que la cobertura de líneas de código no supere el 90% para alguna funcionalidad, se deberá incluir un análisis sobre porque, explicando que faltó en los test para que no se ejercitara la parte de código no cubierto.
   Es necesario que se evidencie a detalle la aplicación de TDD en el requerimiento: **Cálculo de precio de depósito** de tal forma que permita entender como lo lograron. Esto implica que dicha sección debe incluir: casos de prueba a validar y narración de los pasos aplicados (Red, Green y Refactor).
7. Para las funcionalidades de **Administración de Depósitos *y Alta de depósito*** 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, formatos inválidos, pruebas de las reglas del negocio, etc.
   Se debe entregar un reporte que muestre evidencia del resultado de ejecutar estos casos de prueba, que puede ser registrada mediante capturas de pantalla (como anexo en el documento) o realizando uno o más videos cortos publicados en YouTube, en los cuales se pueda apreciar claramente la ejecución de las pruebas. **Los enlaces a estos videos deben incluirse en este documento y se debe verificar que se hayan compartido correctamente y que un tercero pueda verlos.**

Importante: se espera que la descripción y justificaciones sigan un orden lógico, **intercalando diagramas y explicaciones según sea necesario**. Las condiciones de entrega serán evaluadas como si se le estuviese entregando a un cliente real: **prolijidad, claridad, profesionalismo**, etc.

Entrega:

La entrega se realiza por gestion.ort.edu.uy y mediante el repositorio de GitHub de acuerdo con las siguientes instrucciones.

<u>Entrega por Gestión</u>
La entrega de la documentación del obligatorio será en formato digital y por gestion.ort.edu.uy. La misma **debe realizarse antes de las 21 horas** del día de la entrega.

Los principales aspectos a destacar sobre la entrega online de la documentación del obligatorio son:

1. **La entrega de la documentación** se realizará desde gestion.ort.edu.uy (además de estar en el repositorio).

2. Previo a la conformación de grupos cada estudiante deberá estar inscripto a la evaluación. Sugerimos realizarlo con anticipación.
3. Uno de los integrantes del grupo de obligatorio será el administrador del mismo y es quien formará el equipo y subirá la entrega
4. Cada equipo 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)
5. El archivo a subir debe tener un tamaño máximo de 40mb.
6. Les sugerimos realicen una 'prueba de subida' al menos un día antes, donde conformarán el 'grupo de obligatorio'.
7. **La hora tope para subir el archivo será las 21:00 del día fijado para la entrega**.
8. La entrega se podrá realizar desde cualquier lugar (ej. hogar del estudiante, laboratorios de la Universidad, etc)
9. Aquellos que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor pasar por la oficina del Coordinador o por Coordinación adjunta antes de las 20:00hs. del día de la entrega

En caso de tener una situación particular de fuerza mayor, es necesario dirigirse con suficiente antelación al plazo de entrega, a la Coordinadora de Cursos o el Secretario Docente.

Entrega en el repositorio de GitHub

La entrega en el repositorio del equipo en la organización de GitHub (**IngSoft-DA1**.) debe estar compuesta por los siguientes elementos:

- Una carpeta con la aplicación compilada.
- Código fuente de la aplicación y de los tests (merge a main), incluyendo el proyecto que permita probar y ejecutar.
- Documentación digital en PDF (incluyendo modelado UML).
- Release y Tag en la rama main con un nombre adecuado (ver https://docs.github.com/es/repositories/releasing-projects-on-github/managing-releases-in-a-repository). La hora tope para realizar el merge a main y crear el Release / Tag será - **sin excepción** - las 21:00 del día fijado para la entrega.

Rúbrica de evaluación

Puntaje total: 20 puntos.

	Evaluación de	Pts.	Criterios de corrección
Funcionalidad	Implementación de la funcionalidad pedida y calidad de la interfaz de usuario.	6	**Excelente**: Se implementa toda la funcionalidad, tiene buena usabilidad y no hay errores importantes de funcionamiento. **Correcto**: Falta algún punto no central de la funcionalidad, existen detalles no bloqueantes de funcionamiento o la aplicación no es fácil de usar. **No suficiente**: Faltan partes importantes (por alcance o dificultad) de la funcionalidad. Existe gran cantidad de errores o funcionalidades no usables.
Diseño y documentación	Diagramas de paquetes y. tabla de responsabilidades Diagramas de clases Justificación del diseño Calidad del diseño Informe de cobertura de los casos de prueba Evidencia de pruebas funcionales Claridad de la documentación Organización de la documentación (debe tener un orden lógico y un índice) Completitud de la documentación Prolijidad de la documentación	6	**Excelente**: Se presenta un documento completo en función de lo que se pide, ordenado y prolijo, que explica la solución entregada en todos sus aspectos. Se utiliza la notación vista en clase, de manera formal y correcta, para presentar los aspectos importantes del diseño. Se intercalan diagramas y explicaciones, que permiten comprender el diseño de la solución, las decisiones tomadas y la motivación detrás de las mismas. **Correcto:** Se presenta un documento prolijo que describe el diseño de la solución. Se describen los aspectos más importantes, pero hay errores en la nomenclatura. Falta describir aspectos importantes. El orden de la documentación no permite seguir un hilo descriptivo. **No suficiente**: Falta detallar parte importante de la solución. No se usa la notación vista en clase, o se usa en forma equivocada o incompleta en repetidas ocasiones. Se presenta la documentación como una sucesión de diagramas sin explicación.
Calidad del código y metodología de desarrollo	Desarrollo guiado por las pruebas (TDD) y técnicas de refactoreo de código Evidencia de la aplicación de TDD Buenas prácticas de estilo y codificación y su impacto	8	**Excelente**: El código es prolijo, fácil de entender y cumple con los puntos cubiertos en el curso sobre Clean Code. Lo construido coincide con lo detallado en la documentación de diseño. Se evidencia el uso de TDD mediante los commits en el repositorio. Hay buena cobertura de casos de prueba.

	en la mantenibilidad (Clean Code) Correcto uso de las tecnologías Claridad del código Concordancia con el diseño		**Correcto**: El código es prolijo en general, aunque presenta detalles a mejorar. Lo construido se corresponde en términos generales con lo presentado en el documento. Hay pruebas unitarias, aunque no se evidencie el uso de TDD. **No suficiente**: El código es desprolijo o difícil de entender. No cumple en repetidos casos lo propuesto por Clean Code. No hay pruebas unitarias.