Proyecto: “Open Market”

**Tabla de contenido**

[**Entregables 2**](#)

[**Rúbrica de evaluación de cada corte 3**](#)

[**Requerimientos funcionales 5**](#_2et92p0)

[**Valor del proyecto de clase 8**](#_bynikgvkttk8)

El objetivo de este documento es definir los entregables para cada corte del proyecto de clase Open Market de Ingeniería de Software II y Laboratorio de Ingeniería de Software II durante este periodo académico 2023.1. Se toma como referencia el documento de Open Market que contiene todos los requisitos, pero se acota los requisitos a entregar debido al tiempo disponible.

El proyecto Open Market lo deben trabajar en grupos de cuatro estudiantes. Es importante que cada equipo trabaje de forma colaborativa, donde haya buenas relaciones entre todos sus integrantes, donde existan tareas individuales y metas comunes. El equipo debe buscar herramientas que favorezcan la comunicación durante el proyecto y la coordinación de tareas. Se recomiendan: taiga.io, trello o jira.

## **Entregables**

Para los dos primeros cortes se debe entregar (el tercer corte es distinto es una experiencia de desarrollo global):

* Especificación de **requisitos funcionales** (de los más importantes, de los que generen valor al cliente) ya sea mediante escenarios de casos de uso, o Historias de Usuario y criterios de aceptación. Se recomienda antes de implementar una funcionalidad hacer prototipos de la interfaz de usuario (al estilo [wireframe](https://medium.com/rocket-studio-ux/wireframe-mockup-y-prototipos-en-busca-de-sus-diferencias-23a03bcbdb69)s) y aplicar algún método de validación de usabilidad como **thinking aloud**.
* Especificación de **cualidades del software** mediante **escenarios de calidad**.
* Documentación de la arquitectura combinando vistas del modelo C4 y de UML:
  + Vistas de la arquitectura
  + Vista C4
  + Vista de módulos
  + Vista de C&C
  + Vista de Instalación
  + Rationale (Justificación de las decisiones de diseño)
* Prototipo funcional en java.
* Documentar las decisiones de diseño, que tácticas y qué patrones de diseño se aplicaron en la implementación (en un archivo pdf).

***Corte 1:***

* **Primera iteración:** hacer una implementación mediante una arquitectura cliente/servidor monolítica en capas.

***Corte 2:***

* **Segunda iteración:**  hacer otra implementación mediante una arquitectura microkernel y por eventos.

***Corte 3:***

* **Tercera iteración:** hacer una tercera implementación utilizando una arquitectura hexagonal y microservicios. Esta iteración es diferente a los dos anteriores, pues se hará una experiencia de desarrollo global con estudiantes de arquitectura de software de la Universidad Nacional de la Plata - Argentina. Cada equipo de desarrollo se unirá a otro equipo de la UNLP y les ayudarán implementando un microservicio. Los estudiantes de la UNLP serán los encargados de pasar el diseño del microservicio a los estudiantes de Unicauca y guiarlos en la integración y despliegue mediante trabajo colaborativo.

## **Rúbrica de evaluación de cada corte**

Los entregables de cada corte, tanto para la teoría como para el laboratorio, debe cumplir la siguiente rúbrica de evaluación (los elementos resaltados en color gris son obligatorios, si falla alguno, la nota es menor a 3.0):

**Nivel 1: Insuficiente.** Se omiten todos los aspectos resaltados en color gris, casi no se logran las marcas. Nota: 0 – 1.0

**Nivel 2: No alcanzado.** Se omite alguno de los aspectos resaltados en color gris, se alcanzan varias marcas. Nota: 1.1 – 2.9

**Nivel 3: Logrado.** Se alcanzan todos los aspectos resaltados en color gris pero los entregables tienen muchas cosas por mejorar. Nota: 3.0 – 4.0

**Nivel 4: Satisfactorio.** Se alcanza de manera avanzada las características. Cumple todas o la gran mayoría de marcas. Nota: 4.1 – 5.0

**Para el laboratorio:**

| **Característica** | **Descripción** | **Marcas** |
| --- | --- | --- |
| Principios SOLID | Cumple con los principios SOLID. |  |
| Patrón arquitectónico | Se aplica correctamente el patrón arquitectónico solicitado en el corte. |  |
| Patrones de diseño | Se aplican patrones de diseño de forma correcta. |  |
| Requisitos funcionales | Satisface los requisitos funcionales. |  |
| Requisitos no funcionales | Satisface los requisitos no funcionales. |  |
| Código fuente | Indentado perfectamente. |  |
| Documentando con java doc /\*\* \*/, cada clase, método, atributos. |  |
| Sigue las convenciones de java para nombrar paquetes, clases, métodos, constantes. |  |
| Commits de todos los participantes en el repositorio git. |  |
| Obedece al diseño propuesto. |  |
| Pruebas unitarias | Corre satisfactoriamente las pruebas unitarias. |  |

**Para la teoría:**

| **Característica** | **Descripción** | **Marcas** |
| --- | --- | --- |
| Requisitos funcionales | Se especifican correctamente los requisitos funcionales, ya sea como historias de usuarios y criterios de aceptación ó casos de uso. |  |
| Requisitos no funcionales | Se especifican correctamente mediante escenarios de calidad. |  |
| Validación de los prototipos de la interfaz de usuario | Se valida la usabilidad de los prototipos de la interfaz mediante la técnica thinking aloud. |  |
| Decisiones de arquitectura | El tipo de aplicación es justificado: web, web single page, escritorio, móvil, híbrida (web y móvil). |  |
| El estilo arquitectónico es justificado acorde a los requisitos de la aplicación. |  |
| Tecnología elegida, se responden las preguntas: ¿Qué tecnologías ayudan a implementar los estilos arquitecturales seleccionados? ¿Qué tecnologías ayudan a implementar el tipo de aplicación seleccionada? ¿Qué tecnologías ayudan a cumplir con los requisitos no funcionales especificados? |  |
| Gestión del proyecto | Se usa Scrum y se evidencia: backlog, pila del sprint y el burn-down chart. |  |
| Documentación de la arquitectura | Se documenta adecuadamente la arquitectura de la aplicación vistas C4 y UML. |  |
| Diseño detallado | El diseño propuesto involucra patrones de diseño y se utilizan correctamente. |  |

## 

## **Requerimientos funcionales**

A continuación, se especifican algunos requisitos funcionales del proyecto Open Market a manera de **historias épicas**. De cada historia épica pueden salir varias historias de usuario. La historia épica puede ser tomada como el CRUD completo de una funcionalidad grande; luego se deben especificar historias de usuario pequeñas por cada épica para agregar, listar, eliminar, editar, etc.

Es importante que las historias épicas estén ordenadas por valor, es decir, de la más prioritaria a la menos prioritaria para el cliente. Cada equipo de trabajo debe especificar las historias épicas faltantes a este listado.

| **Codigo** | **Historia épica** | **Contexto** |
| --- | --- | --- |
| HE01 | **Yo como** vendedor en la plataforma **necesito** gestionar productos al sistema (CRUD) **para** posteriormente otros usuarios los puedan comprar en línea. | Un producto debe tener al menos nombre, descripción, foto(s), precio, categoría y ubicación (usando la latitud y longitud). |
| HE02 | **Yo como** usuario anónimo y usuario registrado **necesito** buscar productos en el sistema por diferentes criterios **para** detallar los resultados y ver los datos específicos de cada producto. | Las búsquedas se aplican en el título y en la descripción. Se pueden aplicar filtros: por rango de precios, por categoría y por ubicación. |
| HE03 | **Yo como** **usuario registrado** de Open Market **necesito** realizar compras **incluyendo** realizar el pago para que me envien el producto a la casa. |  |
| HE05 | **Yo como** usuario con alguno de los roles del sistema **necesito** iniciar sesión en el sistema **para** acceder a las funcionalidades o servicios que ofrece la plataforma. |  |
| HE06 | **Yo como** usuario anónimo **necesito** registrarme en la plataforma **para** comprar/vender productos en la plataforma |  |
| … | … | … |

Para el **primer y segundo corte** de la asignaturase deben trabajar los siguientes historias de usuario como mínimo (en los dos cortes de trabajan los mismos requisitos funcionales, pero cambia el estilo de arquitectura):

| **Historia Epica** | **Historia de usuario** | **Historia de usuario** | **Contexto** |
| --- | --- | --- | --- |
| HE01 | HU01 | **Yo como** usuario registrado en la plataforma (vendedor) **necesito** *agregar* un nuevo producto al sistema **para** posteriormente otros usuarios lo puedan comprar en línea. | Un producto debe tener al menos nombre, descripción, foto(s), precio, categoría y ubicación ( usando la latitud y longitud). |
| HE03 | HU02 | **Yo como** usuario registrado de Open Market necesito (comprador) agregar un producto al carrito de compras **para** posteriormente realizar el pago y que me envien el producto a la casa. | Al momento de agregar el producto al carrito se debe solicitar la cantidad a comprar. |
| HE03 | HU03 | **Yo como** usuario registrado (comprador) de Open Market **necesito** visualizar los ítems que he agregado al carrito de compras **para** posteriormente realizar el pago y que me envien el producto a la casa. | Al visualizar los ítems del carrito, se debe mostrar el valor total de la compra. Se pide listar los ítem por página de 5, 10, 20, 50 elementos. |
| HE04 | HU04 | **Yo como** entregador o repartidor de productos necesito entregar el producto y cambiar el estado a “entregado” **para** finalizar el proceso de compra | Una vez el repartidor entrega el producto, entra al sistema y coloca datos de la entrega: fecha, persona que lo recibió, y el estado del producto cambia a *Entregado*. |

Para el tercer corte se hará la experiencia de desarrollo global con los estudiantes de la UNLP. Solamente se debe entregar el microservicio implementado y probado.

## **Valor del proyecto de clase**

A continuación los valores del proyecto de clase respecto a las dos asignaturas en cada corte.

Laboratorio de Ingeniería de Software:

* Talleres: 50%
* Proyecto de clase: 50%

Ingeniería de software II:

* Proyecto de clase: 25%
* Talleres: 15%
* Parcial: 60%