Laboratorio IngSoft1 - 2024

Plataformas de trabajo

Cada grupo debe crear:

1. Una organización de Github (no repos sueltos en sus cuentas), con un repositorio para la aplicación back y uno para el front.
2. Un proyecto de JIRA (**debe ser para proyectos SCRUM**)
3. Un canal **privado** de Zulip con el nombre IngSoft1-<nombre_del_equipo>

Ejemplo: `IngSoft1-EquipoDeProfes`

Luego deben invitar a ambos profes a las tres plataformas (los profesores de su equipo les darán sus usuarios de JIRA / Github)

Metodología

1. Se utilizará SCRUM
2. Cada grupo se puede organizar internamente, pero todos los integrantes deben estar al tanto del diseño, definición de API, estructura del código, procedimientos, etc.
3. Ejecutaremos 3 Sprints de 3 semanas cada uno, incluyendo planning, daily, demo, retro.
4. En cada uno de estos Sprints, cada equipo debe definir un PO y un SM, (sin repeticiones entre sprints)
    a. Product Owner (PO): es el encargado de la comunicación entre los clientes (profes) y el resto del equipo.
    b. Scrum Master (SM): es el responsable de que el equipo siga el proceso de trabajo, maneja y facilita las reuniones, se asegura de que los objetivos del sprint se cumplan.
5. Se deben implementar 2 aplicaciones completamente independientes: un backend que implementa el manejo de datos y la lógica de negocio, y un frontend que maneja la interacción con los usuarios.
6. Se debe proveer la definición de API que comunicara estas aplicaciones.

Metodología

Sprints

1. Duran 3 semanas (incluye la última clase, en la cual se hace la demo, y la retro)
2. Los profes darán el listado de requerimientos esperados para cada sprint.

Planning

1. Cada equipo debe descomponer los requerimientos en tantos issues cómo crean necesarios para completarlos
2. Se debe estimar cada uno de estos issues
3. Se deben crear todos los issues en JIRA, definiendo desarrollador y uno o más reviewers asignados.
4. Se debe definir el compromiso (objetivo) del sprint (sprint backlog), dependiendo de la capacidad del equipo.
5. Además de los nuevos requerimientos, se pueden arrastrar issues de los sprints anteriores (carry-over). **Éstos siempre toman precedencia sobre los requerimientos nuevos.**

Metodología

Compromiso Sprint 2:

Obligatorio:

- Carry-over Sprint 1:
  - Bugs de Lobby solucionados
  - Mezclar carta, repartir manos, sortear e informar de turno
- Listar partidas
- Abandonar partida desde lobby
- Terminar partida al quedar solo un jugador
- Jugar carta lanzallamas

No compromiso pero sería muy bueno para el equipo:

- Descartar carta

Extra (nice to have):

- Completar el diseño de los websockets para el intercambio de cartas, empezar con la implementación si es posible

Metodología

Daily

1. La tendrán con alguno de sus profesores asignados cada día de clase
2. Es recomendable hacerla entre ustedes los off-days

Retro

1. La hace cada grupo internamente luego de la demo
2. Los resultados y action items deben ser posteados en el canal de cada grupo

Metodología

Demo

Es la entrega en la cual el cliente evaluará el avance logrado en el sprint. **Es equivalente a la evaluación del lab**

Cada grupo debe, **el dia anterior a la demo**:

1. Enviar por zulip un listado de todos los requerimientos que se entregarán en la demo (requerimientos que se consideran completados)
2. Enviar por zulip un listado de todos los requerimientos que no pudieron ser completados
3. Enviar por zulip un video, grabado en Loom, mostrando la funcionalidad de los requerimientos completados

Durante la clase, cada grupo contará como máximo con 20/30 minutos para presentar la demo. Esto incluye setup, revisión del loom, preguntas y feedback de los profesores. Coordinan los horarios asignados a cada grupo con sus profes asignados.

Herramientas y Tecnología

1. Se utilizará Fast API para el backend, y React para el frontend.
    a. Puede utilizarse cualquier librería que el equipo considere necesario
    b. Es necesario implementar una conexión por websockets para la actualización de datos entre múltiples clientes

La recomendación de la materia es:
Front -> Back, a través de HTTP REST
Back -> Front, a través de WS Websockets

Los websockets no son opcionales

Herramientas y Tecnología

JIRA

1. No es necesario crear EPICS, pero pueden hacerlo para mantener mejor organización.
2. Crear un issue de tipo STORY por cada ticket resultante en la descomposición (no por cada user story). De esta forma es más sencillo trackear exactamente lo que está haciendo cada miembro del equipo. Los issues de tipo TASK son para tareas no referidas al código. Idealmente, una STORY no debería tener sub-issues.
3. Utilizar un formato de escritura estructurado given-when-then
4. Escriban bien los tickets. Se debe entender exactamente qué se pretende implementar en cada uno, y que se debe entregar. Usen Acceptance Criteria. Deben estar redactados de manera formal, y **deben poder ser comprendidos por un desarrollador de otro equipo**. Los issues pueden incluir todos los detalles técnicos que crean necesarios. **Tomense el tiempo para escribir buenos issues**
5. Se debe asignar desarrolladores a cada tarea. Cada tarea debe estar estimada. Integren Github a JIRA asi pueden linkear branches y PRs
6. Su workflow debe consistir de, como mínimo: TODO, IN PROGRESS, IN REVIEW, DONE.

Herramientas y Tecnología

Título:

Implementar lobby de partida en espera

Descripción:

Dado que un usuario necesita obtener información sobre la partida a la cual se ha unido, mientras esta se encuentra en estado no-iniciada (esperando jugadores / esperando que el owner inicie la partida), **cuando** un jugador se une exitosamente a una partida disponible (ya sea pública o privada), **entonces** se debe redirigir a una nueva interfaz (lobby), donde puede obtener información sobre la partida.

Esta información será:

- El nombre de la partida
- Su tipo (pública / privada)
- Los jugadores que están en espera (incluido el owner, que debe estar resaltado con un icono especial)
- El botón de iniciar partida, en caso de que el usuario sea owner de la misma
- El botón de “abandonar partida” (o “cancelar partida” si el usuario es owner)

Herramientas y Tecnología

Tech Notes:

- Ya tenemos un componente genérico que lista elementos en columnas (ItemList), re-utilizarlo en esta interfaz. El fondo, colores y fonts debe ser consistente con la pantalla principal (Home). Utilizar nuevo slug de router: `/lobby/<partida_id>`
- Este componente debe estar conectado al websocket para obtener actualizaciones
- Dividir componentes en `container / components`. Las interacciones con la API serán: Abandonar / Cancelar partida, Iniciar partida.

Acceptance Criteria:

- Nueva pantalla (lobby), donde se muestra la información requerida. Los jugadores son redireccionados a esta pantalla al unirse a una partida
- Debe obtener actualizaciones sobre jugadores unidos y inicio de partida desde el WS
- Debe permitir a los jugadores abandonar (cancelar). Debe permitir al owner Iniciar
- Todos los componentes deben mantener modularización, tener cobertura de tests

Herramientas y Tecnología

Definición de DONE

Un ticket solo se mergea desde su feature branch a develop, cuando se considera DONE.
La definición de DONE será:

1. El codigo esta terminado y completo, cumple con la Acceptance Criteria del ticket
2. El código ha sido revisado y aprobado por al menos 1 desarrollador que no estuvo involucrado en su implementación
3. El código **incluye los unit test** necesarios para testear automáticamente sus casos base, casos de éxito, excepciones, y corner cases.
4. Esto quiere decir: los UT deben ser implementados como parte del ticket. **No en un ticket separado**

Herramientas y Tecnología

GitHub

El equipo debe tener una organización de github, con 2 repositorios: uno para back, uno para front. Cada repo debe tener un README en el root, **que indique de manera clara y completa como configurar y correr la aplicación**

Pueden agregar una wiki a la org / repo con la documentación relevante (TH, API)

Se debe utilizar la convención master-develop-feature para el manejo de branches:

1. Cada issue nuevo (feature o bugfix), se trabaja en su propia branch
2. Luego de que el código es revisado y aprobado (incluyendo sus UTs), es decir, cuando el issue puede ser considerado DONE, ese branch se mergea a develop
3. Antes de cada demo en el final del sprint, se hace un “release”: se mergea el current develop a master, y se crea un tag (v1, v2, v3) en máster, que indica el contenido del sprint. Las demos se hacen desde máster.
4. No pedimos Squash and Merge. Aprendan a solucionar conflictos. Mantengan las ramas el tiempo adecuado de vida

Herramientas y Tecnología

Utilicen convención de nombres:

1. Cada branch debe llamarse: (feature | bugfix)_<ID de issue>_<nombre>.
   Ejemplo: `feature_ING-16_get_games_endpoint`
2. Los commits deben empezar con el id del issue:

Ejemplo:

```
ING-16 add GET games endpoint
ING-16 add query to game service
ING-16 add tests for new endpoint
```

Evaluación

1. Se evaluará todo el proceso de desarrollo, no solo los entregables. La nota del laboratorio es individual.
2. Todo el proceso del lab, incluyendo organización del equipo, participación en las reuniones, contribución de código es evaluado
3. Aun cuando el código no define la nota, **no es posible aprobar** si no se entrega lo pedido oportunamente por el profesor o su replanteamiento aprobado por los profesores respectivos.
4. Entregar el último día un código al 100% no los aprueba.
5. Seguir perfecto el proceso y no tener un código funcional tampoco.
6. El lab tiene 3 entregas (las demos) y una defensa final (oral).
7. Cada demo cuenta como una entrega o parcial. **Para el lab, no venir a una demo es equivalente a no venir a un parcial del teórico/práctico.**

Sprint 1 - 10/09 a 01/10

Martes 10/09

- Les damos la lista de requerimientos para el sprint 1
- Deben trabajar grupalmente en la descomposición y estimación
- Deben crear las cuentas necesarias, canal de zulip, etc

Jueves 12/09

- Tener la descomposición y estimación lista
- Todo en JIRA
- Revisión de la descomposición, planning junto a su profe asignado

Martes 01/10 (Demo)

- **Lunes 30/09 Deben tener listo el video en loom y los requerimientos completados el lunes anterior**
- Demo con su profesor asignado

Sprint 1 - 10/09 a 01/10 -

- Crear Partida (pública)
- Listar Partidas
- Unirse a Partida
- Abandonar Partida (NO CANCELAR)
- Iniciar Partida
- Obtener Movimientos
- Obtener cartas de Figura
- Ver mis propias cartas
- Ver el Tablero
- Ver Información del turno
- Terminar Turno
- Ganar (por abandono)