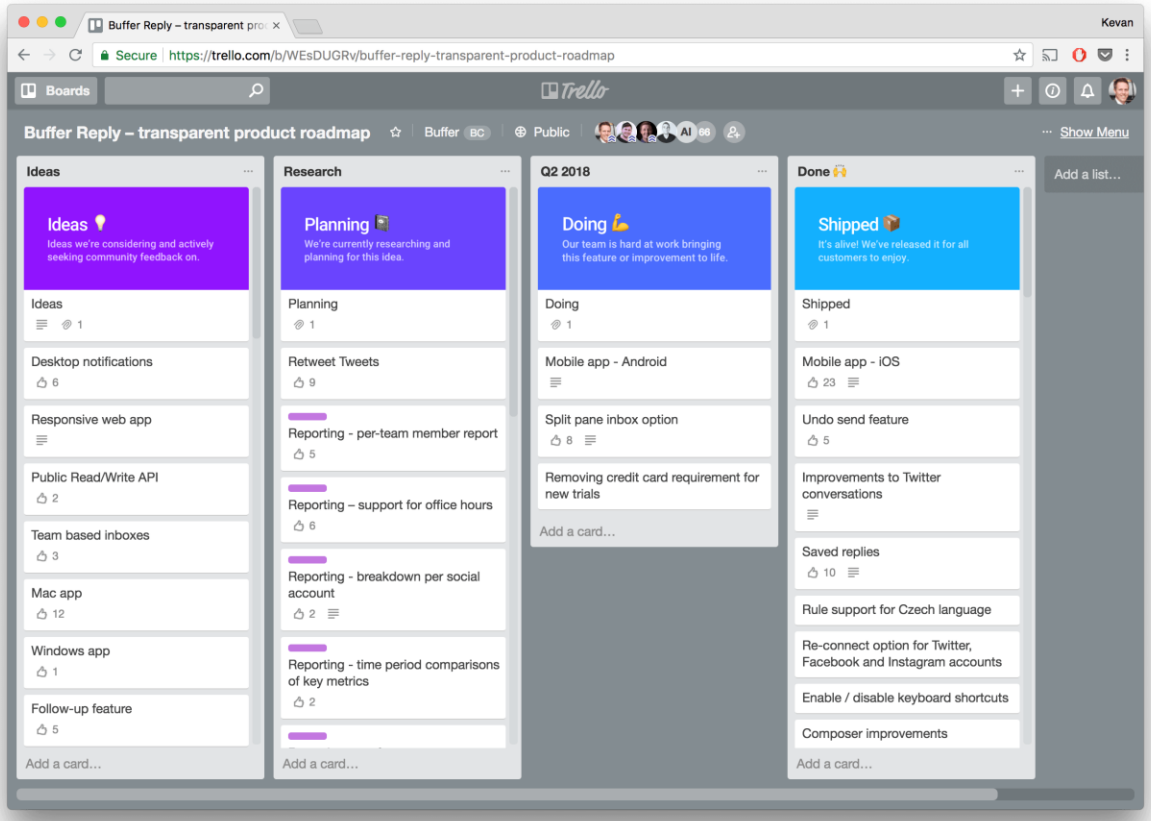
Presentación de las prácticas

Prácticas PDS

Proyecto

Proyecto

- Aplicación de gestión de proyectos
  - Tableros y tareas
  - Inspirado en Trello (https://trello.com/)



Proyecto

- Funcionamiento básico
  - Un usuario puede crear un tablero simplemente indicando su dirección de correo electrónico
    - Esto generará un enlace único y público para acceder al tablero
  - En un tablero se pueden añadir listas de tareas
  - En las listas de tareas se pueden añadir tarjetas que representan tareas
  - Las tarjetas pueden ser de diferente tipo (ej., tareas simples o *check* lists)
  - Las tarjetas tienen etiquetas de diferentes colores para poder identificarlas fácilmente
  - Las tarjetas se pueden mover entre las listas
    - El sistema de registrar la historia de las tarjetas
  - Los tableros se pueden bloquear durante un tiempo dado, de manera que no se pueden crear nuevas tarjetas, solo moverlas.

Proyecto

- Características opcionales

Fácil Implementar reglas a nivel de tablero o de lista de tareas:
  - Una lista no puede tener más de N ítems (configurable)
  - Una lista define que una tarjeta tiene que haber pasado por otra listas antes de llegar a ella.

Fácil Filtrado de tarjetas por etiquetas

Fácil Creación de plantillas para tableros.
  - Las plantillas se definen como un fichero YAML.

Medio Compactación automática de tableros.
  - Archivo automático de tarjetas (ej., tarjetas no completadas en más de una semana).

Medio Autenticación basada en código por correo.

Difícil Permisos para usuarios.

Difícil Reglas de automatización.
  - "Si ocurre evento X -> hacer Y".

- Alguna otra característica comentada con el profesor de prácticas

Proyecto – Requisitos de implementación

- Lenguaje de programación Java.
- Arquitectura Hexagonal siguiendo DDD
- Backend implementado en SpringBoot
- Interfaz gráfica de escritorio con JavaFX
- Maven como sistema de construcción.
- Persistencia con JPA.
- Pruebas de software.

Organización

Organización

- Equipos de 3 estudiantes
  - Se permiten equipos con integrantes de diferentes subgrupos
  - Ejemplo: un grupo formado por estudiantes de los subgrupos 1.2, 2.1 y 3.3
- Los equipos se formalizarán a través de un formulario que se enviará a través del AV
  - Tras este proceso se publicará un listado de los equipos y del profesor responsable de cada equipo
- Cada equipo deberá crear un repositorio privado en GitHub
  - Se dará de alta al profesor responsable invitándolo al repositorio
- Todos los artefactos del proyecto, incluyendo la documentación deben estar alojados en el proyecto de GitHub.
  - Se deberá utilizar GitHub para la gestión del proyecto.

Evaluación

- Diseño del sistema y su justificación.
  - Las clases deberán tener comentarios relevantes explicando el diseño.
- Breve documentación donde se describa el modelo DDD obtenido y las decisiones de diseño
  - Evitar crear diagramas ilegibles
- Corrección de la implementación y otros atributos de calidad (ej., mantenibilidad, legibilidad, extensibilidad, etc.)
- Bueno uso del lenguaje Java
- Bueno uso de las pruebas de software y la cobertura del código obtenida, en particular de la parte del modelo de dominio.
- Organización y limpieza del repositorio

Evaluación - Entregable

- En la asignatura el entregable es el repositorio: ¡cuídalo!
- Usar Markdown para los documentos (.md)
  - Para que se puedan revisar
- Ejemplo de organización
  - `requisitos`
    - `historias-usuario`
    - `pantallas`

    Opcional , solo si es útil.
    Se pueden escribir en *issues* (marcarlos con la etiqueta historia-usuario)
    Se pueden documentar directamente en el código
  - `diseño`
    - `modelo`

    Explicación desde el punto de vista DDD.
  - `documentacion:`
    - `manual:` Manual de usuario de la aplicación
    - `dev:` Información para desarrolladores
  - `java:` proyecto Maven

Trabajo en equipo

- Todos los componentes deben participar en el proyecto
- La división de tareas puede realizarse como se prefiera
  - División por características
    - Más realista, pero más complicada de llevar a la práctica
    - Todos los miembros del equipo implementan todas las capas
  - División por componentes de la implementación
    - Un miembro se encarga del modelo de dominio, otro de la API REST, etc.
  - Combinaciones de ambas
- En cualquier caso, TODOS los miembros deben conocer el proyecto y demostrarlo

Trabajo en equipo

- Entrevistas
  - A criterio del profesor correspondiente
  - Preguntas sobre la práctica
  - Ejercicio de programación
- No superar la entrevista implica suspender el proyecto

Trabajo en equipo

- Fichero CREDITOS.md
- Información sobre en qué ha participado cada miembro de equipo
- Cómo se ha repartido el trabajo
- Mostrar ejemplos/testigos de esto:
  - Commits relevantes
  - PRs y revisiones de código
  - Issues y discusiones
  - Explicación
- No se pretende una descripción exhaustiva pero debe quedar clara la participación de todos los miembros del equipo.