Grupo 29

Item 1. Changelog

Khawla Al Asfar

Juan Carlos Utrilla Martín

Juan Rodríguez Dueñas

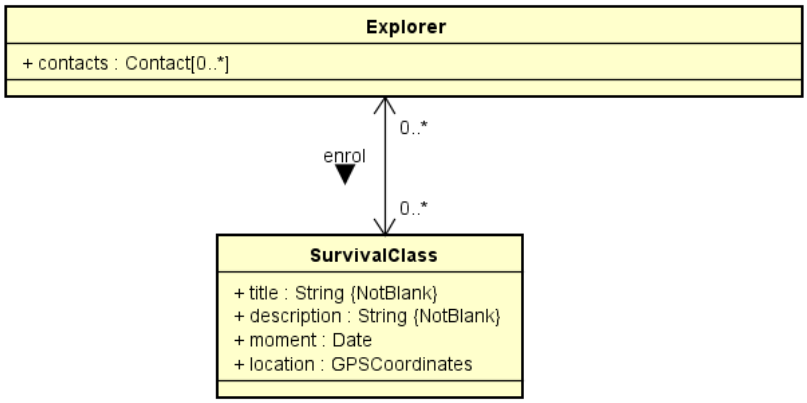
Yassine Taziny

José Manuel Lara Morilla

Contenido

1. Cambios en los diagramas

- **Explorer - SurvivalClass**: añadida la relación entre Explorer y SurvivalClass ya que un Explorer puede inscribirse en un SurvivalClass siempre que tenga una solicitud (application) asociada a un Trip esté aceptada (Status = ACCEPTED).



- Actor:
    - Añadida la restricción en el campo address a obligatorio. Este cambio se indicó en el entregable D03, pero no se incluyó.

2. Cambios en el modelo de dominio

- **Explorer.java**:
    - Por la modificación añadida en el diagrama, hemos añadido la relación con SurvivalClass y sus correspondientes métodos get y set.

```
// Relationships ----------------------------------------------------------

private Collection<SurvivalClass>   survivalClasses;


@NotNull
@Valid
@ManyToMany(mappedBy = "explorers")
public Collection<SurvivalClass> getSurvivalClasses() {
    return this.survivalClasses;
}

public void setSurvivalClasses(final Collection<SurvivalClass> survivalClasses) {
    this.survivalClasses = survivalClasses;
}
```

- **SurvivalClass.java**:
    - Por la modificación añadida en el diagrama, hemos añadido la relación con Explorer.

```
// Relationships ----------------------------------------------------------

private Collection<Explorer>    explorers;

@NotNull
@Valid
@ManyToMany
public Collection<Explorer> getExplorers() {
    return this.explorers;
}

public void setExplorers(final Collection<Explorer> explorers) {
    this.explorers = explorers;
}
```

- **Actor.java**:
    - **getAddress()**: añadida la etiqueta **@NotBlank**.

3. Cambios en los repositorios y servicios

- **Clase Trip**:
  - **TripRepository.java**:
    - **findAvailableTrips()**: añadido el método que devuelve todos los Trips disponibles.
  - **TripService.java**:
    - Añadida la anotación **@Transactional**
    - **create()**: quitada la inicialización de los finders, ya que cortamos la navegabilidad de trip a finder por no ser necesaria.
    - **findAvailableTrips()**: añadido el método que devuelve todos los Trips disponibles.
  - **TripServiceTest.java**:
    - **testDelete()**: modificada la creación de la fecha, en vez de utilizar Timestamp cambiamos a Date.
    - **testFindAvailableTrips()**: añadido el método que comprueba si recibe un conjunto de Trips disponibles respecto al momento actual (si la fecha de comienzo (startTripDate) es mayor o igual que el momento actual (moment)).
    - **assignedCategoryToTrip()**: ha raíz de añadir un método en category para comprobar que un hijo no tenga el mismo nombre que su padre ni que los hijos de este padre, para verificar que funciona correctamente hemos creado una nueva categoría y asignado un padre con hijos para así comprobar que el método de asignar funciona correctamente y comprueba las restricciones de forma correcta.
    - **testAvgMinMaxDevApplicationsPerTrip()**: añadido la comprobación Assert para verificar que se realiza correctamente.
    - **testAvgMinMaxDevPriceOfTheTrips()**: añadido la comprobación Assert para verificar que se realiza correctamente.
    - **testRatioOfTripsCancelledVsTotalTripsOrganized()**: añadido la comprobación Assert para verificar que se realiza correctamente.
    - **testListingTrips10PercentMoraApplicantionsThanAvg()**: añadido la comprobación Assert para verificar que se realiza correctamente.
    - **testMinMaxAvgDevNotesPerTrip()**: añadido la comprobación Assert para verificar que se realiza correctamente.
    - **testMinMaxAvgDevAuditRecordPerTrip()**: añadido la comprobación Assert para verificar que se realiza correctamente.
    - **testRatioOfTripsWithAnyAuditRecord()**: añadido la comprobación Assert para verificar que se realiza correctamente.
    - **testAuditorPerTrip()**: añadido la comprobación Assert para verificar que se realiza correctamente.
- **Clase Audit**:
  - **AuditServiceTest.java**:
    - **testDeleteByAuditor()**: añadida una línea para que después de realizar el método el usuario salga del sistema.
    - **testFindByAuditor()**: añadida una línea para que después de realizar el método el usuario de desloguee del sistema.

- **Clase Configuration**:
    - **ConfigurationServiceTest.java**:
        - **testSave()**: añadida una línea para que después de realizar el método el usuario de desloguee del sistema.
- **Clase Story**:
    - **StoryServiceTest.java**:
        - **testAssignStoryToTrip()**: añadido la desautenticación en el método.
- **Clase Category**:
    - **CategoryRepository.java**:
        - **existsThisCategoryName(...)**: añadido el método
    - **CategoryService.java**:
        - **save()**: añadida la restricción de que el nombre de la categoría no sea el mismo que el nombre de la categoría padre.
- **Clase Manager**:
    - **ManagerServiceTest.java**:
        - **testCreate()**: añadida la autenticación de admin.
    - **testSave()**: añadida la autenticación de manager (también valdría para admin).
    - **testDelete()**: añadida la autenticación de admin.

- **Clase SurvivalClass**:
    - **SurvivalClassRepository.java**:
        - **findByExplorer(...)**: creado este método que devuelve todos los SurvivalClass relacionados con un Explorer.
        - **checkEnrolExplorerToSurvivalClass()**: creado este método que devuelve si el Explorer que se quiere apuntar (enrol) a un SurvivalClass tiene alguna solicitud (Application) aceptada (Status = ACCEPTED).
        - **isEnrolExplorerToSurvivalClass()**: creado este método que devuelve si un Explorer está apuntado (enrol) a un SurvivalClass.
    - **SurvivalClassService.java**:
        - **enrolAnExplorerToSurvivalClass()**: método que asigna un survivalClass a un Explorer.
        - **disenrolAnExplorerToSurvivalClass()**: método que desasigna un survivalClass a un Explorer.
        - **checkEnrolAnExplorerToSurvivalClass()**: método que devuelve un Boolean comprobando si un Explorer puede apuntarse (enrol) en una SurvivalClass.
        - **checkDisnrollAnExplorerToSurvivalClass()**: método que devuelve un Boolean comprobando si un Explorer puede desapuntarse (unenrol) en una SurvivalClass.
        - **checkByPrincipalExplorer()**: añadido el método para que devuelva si el actor logueado o no es un Explorer.
        - **checkByPrincipalManager()**: modificado el método para que devuelva si el actor logueado o no es un Manager.
        - **save(...)**: modificado el método para que distinga si el actor es Manager o Explorer. Dependiendo de quien sea, realizará unas operaciones u otras.

        - **findByTrip(...)**: corregido el método que devuelve el objeto SurvivalClass.
        - **findByExplorer(...)**: añadido el método que devuelve todos los survivalClass relacionados a un Explorer.
        - **deleteByExplorer(...)**: añadido el método que devuelve que elimina todos los survivalClass relacionados con un Explorer.
    - **SurvivalClassTest.java**:
        - **testSave(...)**: hemos dividido este método en dos:
        - **testSaveExplorer(...)**: verifica los cambios cuando un usuario de tipo explorer está logueado en el sistema.
        - **testSaveManager(...)**: verifica los cambios cuando un usuario de tipo manager está logueado en el sistema.
        - **testDeleteByManager()**: eliminada línea duplicada this.survivalClassService.deleteByManager(manager);
        - **testFindByTrip()**: añadido una comprobación del elemento a encontrar.
        - **testDeleteByManager(...)**: modificado el método para que valide que los datos se han eliminado.
        - **testFindByManager(...)**: modificado el método para que valide que los datos se han recbido correctamente.
        - **testCheckEnrolAnExplorerToSurvivalClass()**: añadido el método que valida si un Explorer puede apuntarse (enrol) a un SurvivalClass.
        - **testCheckDisenrolAnExplorerToSurvivalClass()**: añadido el método que valida si un Explorer puede desapuntarse (disenrol) a un SurvivalClass.
- **Clase Application**:
    - **ApplicationRepository.java**:
        - **findApplicationByStatus(Status status)**: creación del método.
    - **ApplicationService.java**:
        - **save()**: modificado el método save para que retrase un segundo el momento en el que un application persiste en la base de datos.
        - **acceptApplication(...)**:
            - Modificación del método para que devuelva el objeto devuelto por el método save (Application application).
            - Añadidas comprobaciones de que creditCard recibida como parámetro no tiene ninguno de sus atributos a nulo.
        - **rejectApplication(...)**: modificación del método para que devuelva el objeto devuelto por el método save (Application application).
        - **dueApplication(...)**: modificación del método para que devuelva el objeto devuelto por el método save (Application application).
        - **cancelApplication(...)**: modificación del método para que devuelva el objeto devuelto por el método save (Application application).
    - **ApplicationServiceTest.java**:
        - Reorganización general del código que no supone ningún cambio funcional de la aplicación.
        - En lugar de buscar por un id de una application, vamos a buscar una aplicación cualquiera cuyo Status sea PENDING.

        - **testAcceptApplication()**: creación del método. Cuando se utiliza el método acceptApplication de applicationService, este devuelve el application modificado que ha persistido para su posterior comprobación. El problema es que no hemos recogido el application persistido en la base de datos. El error ha sido corregido. Lo mismo para **testDueApplication(), testRejectApplication().**
        - **testRejectApplication()**: creación del método

- **Clase Explorer**:
    - **ExplorerService.java**:
        - **create()**: como se ha incluido la relación Explorer (0..*) - SurvivalClass (0..*) en el modelo, hemos inicializado el atributo survivalClass de Explorer (que es una lista).
        - **delete()**: como se ha incluido la relación Explorer (0..*) - SurvivalClass (0..*) en el modelo, se ha añadido una línea para eliminar todos los SurvivalClass asociados a un Explorer.

- **Clase Note**:
    - **NoteService.java**:
        - **replyToNote()**: añadido Assert.isNull(note.getReply()) en el método que comprueba que la nota todavía no ha sido respondida.
        - **save()**: añadida una comprobación "if" que permite a los dos actores manager y auditor realizar la operación de guardar.
    - **NoteServiceTest.java**:
        - **testReplyToNote()**: reemplazado el método Assert para que compruebe el elemento persistido en la base de datos.
        - **testFindByAuditor()**: añadido el assert para comprobarlo.
- **Clase ProfessionalRecordService**:
    - **ProfessionalRecordService.java**:
        - **save()**: en el método save guardamos primero el professionalRecord luego lo añadimos a la lista de professionalRecords del curriculum.
    - **ProfessionalRecordServiceTest.java**:
        - **testDelete()**: añadida la etiqueta @test
- **Clase PersonalRecordService**:
    - **PersonalRecordRepository.java**:
        - Añadida la etiqueta **@Repository**.
    - **PersonalRecordService.java**:
        - **delete()**: eliminada la referencia al PersonalRecord del curriculum asociado
- **Clase Sponsorship**:
    - **SponsorshipServiceTest.java:**
        - **testFindBySponsor():** añadido la comprobación Assert para verificar que se realiza correctamente
- **Clase LegalText**:
    - **LegalTextServiceTest.java:**
        - **testFindLegalTextperTrips():** añadido la comprobación Assert para verificar que se realiza correctamente