ENTREGA 4
CAMBIOS UML Y JAVA

Diseño y Pruebas

Grupo 20:
José Ángel Domínguez Espinaco
Daniel Lozano Portillo
José Joaquín Rodríguez Pérez
María Ruiz Gutiérrez
Miguel Ternero Algarín
Laura Vera Recacha

Índice

Introducción

En este documento serán descritos todos los cambios de nuestro Modelo de Dominio y Modelo Java, los cuales irán acompañados de capturas.

Cambios realizados UML

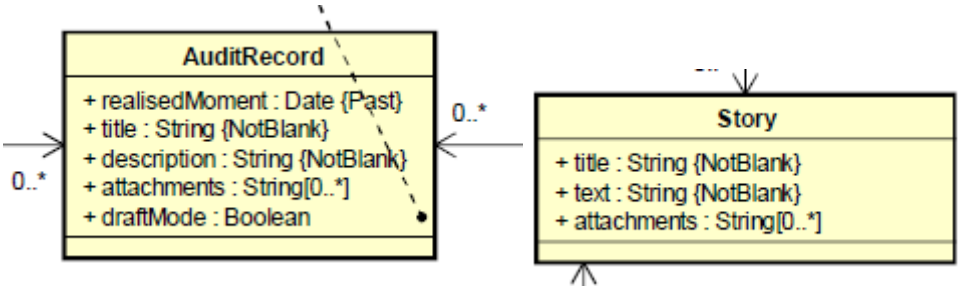
1. En la clase Sponsorship el atributo bannerURL puede ser opcional por lo tanto hemos cambiado la multiplicidad a [0..1].

Antes:

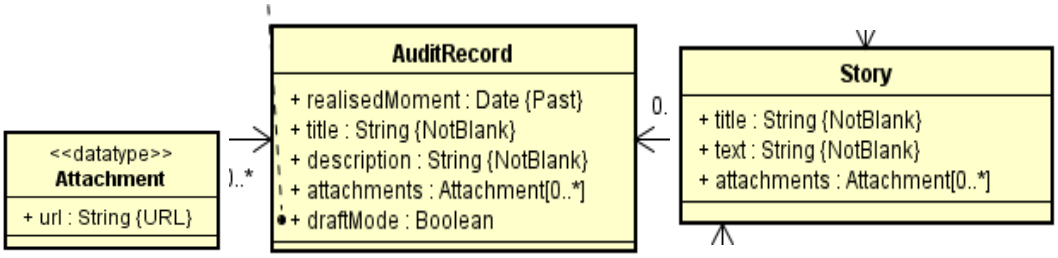
Después:

2. En las clases Story y Auditrecord se ha cambiado el tipo String del atributo attachments, creándose una clase datatype Attachment ya que al estar puesto como colección de String no podíamos poner una restricción de tipo @URL y al crear el datatatype ahora es ahí donde establecemos la restricción @URL.

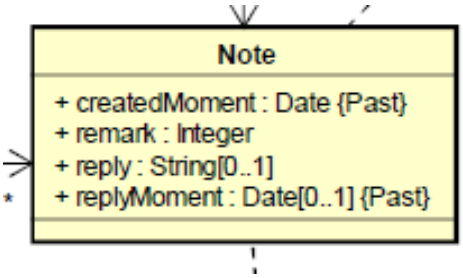
Antes:



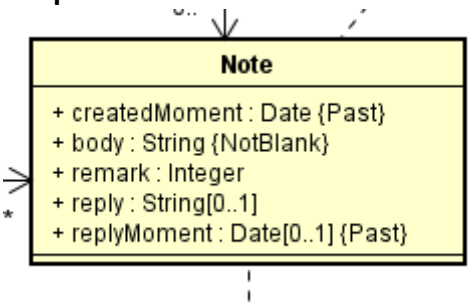
Después:



3. En la clase Note hemos añadido el atributo body de tipo String ya que es necesario a la hora de escribir el contenido de una nota.
Antes:



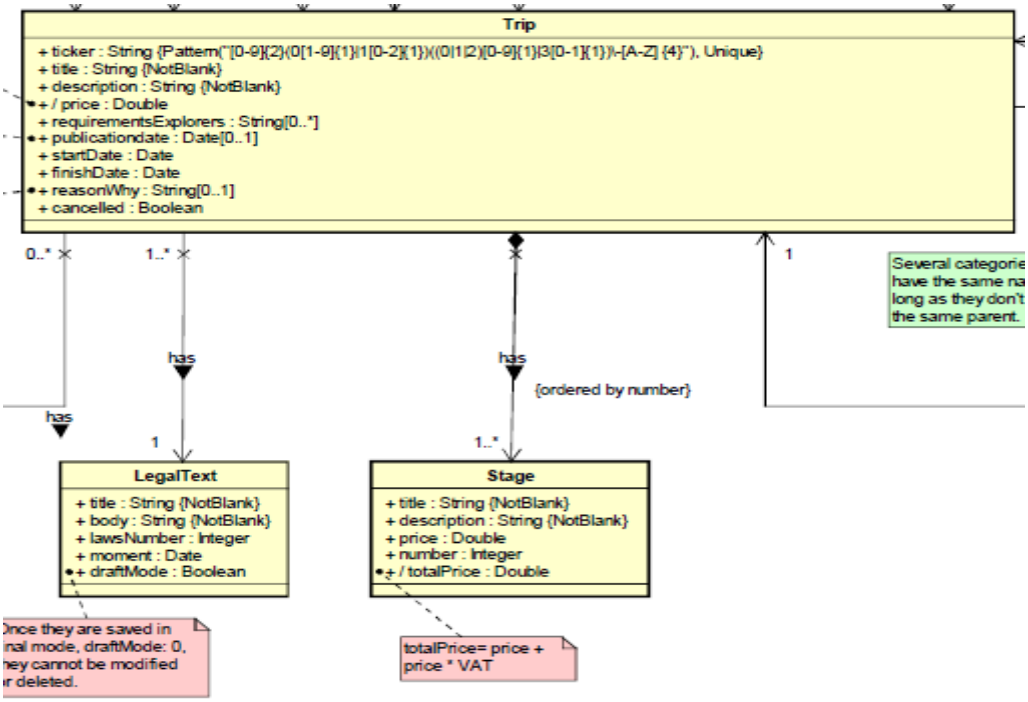
Después:



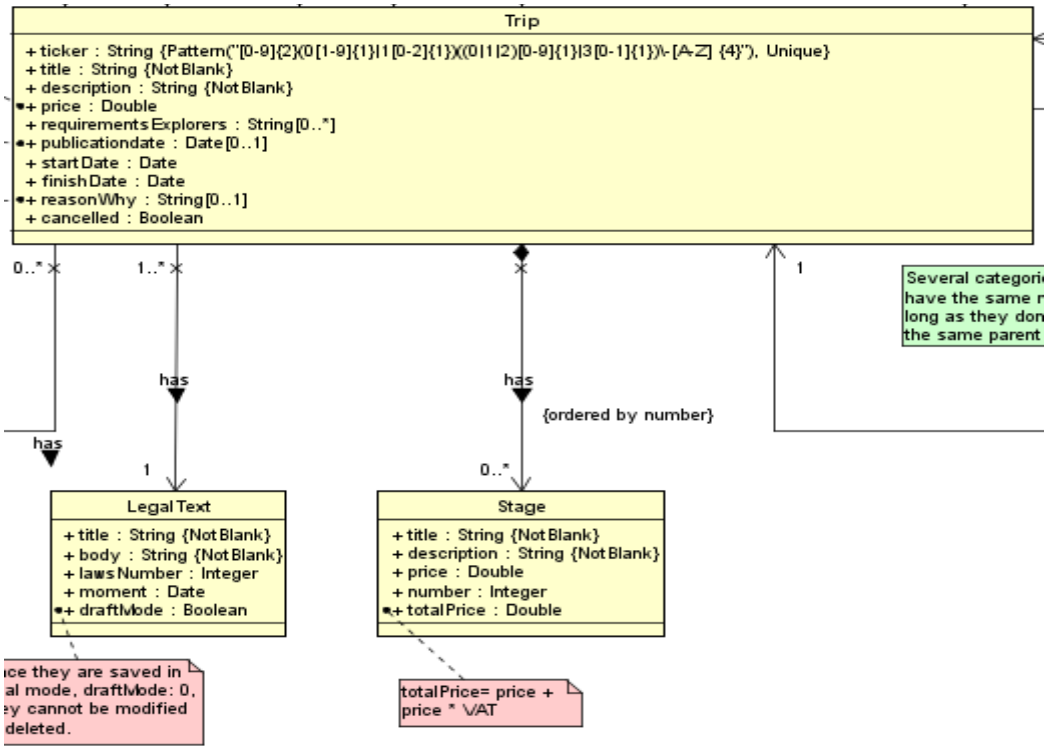
4. En la asociación de Trip con Stage hemos cambiado la multiplicidad a [0..*] ya que puede existir un Trip que aún no tenga ninguna stage asiganada.

En las clases Stage y Trip sus atributos derivados totalPrice y price pasan a ser básicas ya que se calcularán en un método en los servicios StageService y TripService respectivamente.

Antes:

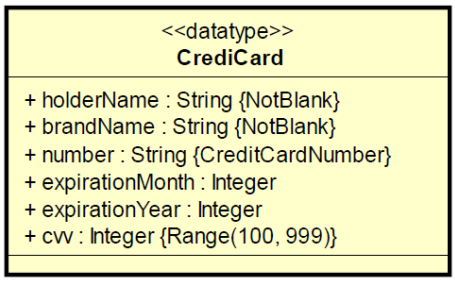


Después:

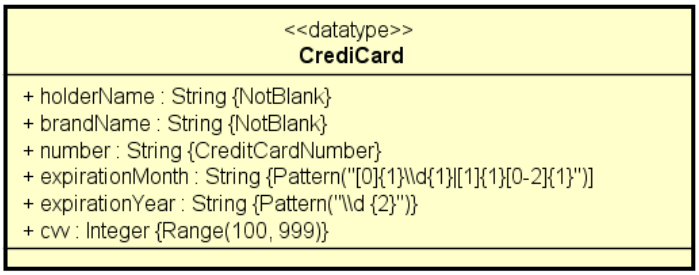


5. En CreditCard hemos cambiado los tipos de los atributos expirationMonth y expirationYear por String para comprobar sus restricciones mediante patrones que también se han añadido.

Antes:

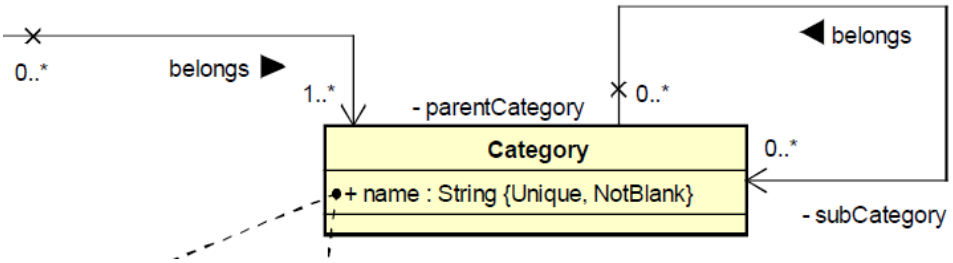


Después:

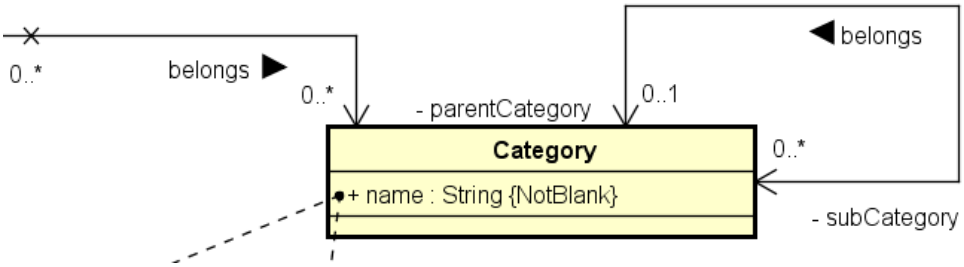


6. En la clase Category hemos cambiado la multiplicacidad de la asociación con trip que pasa a ser [0..*] porque cuando creamos un Trip no tenemos por qué asignarle una categoría y la asociación y navegabilidad de una parentCategory que pasa a ser bidireccional y con multiplicidad [0..1].

Antes:

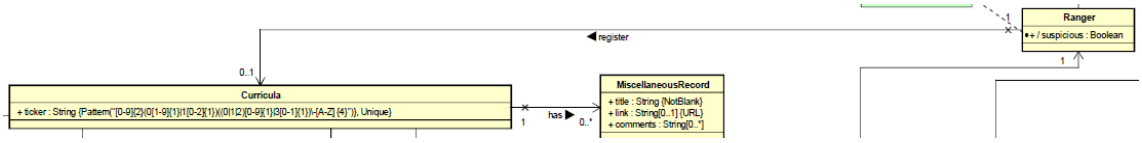


Después:

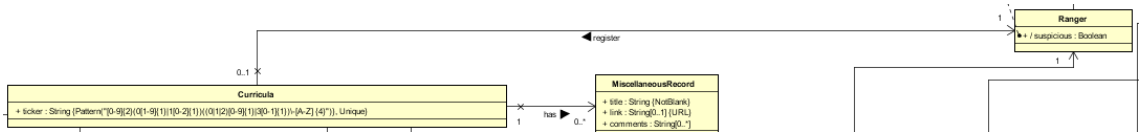


7. En la asociación de la clase Ranger con la clase Curricula hemos cambiado la navegabilidad al contrario, siendo ahora de Ranger hacia Curricula porque no puede existir una Curricula sin asignarle ningún Ranger pero sí puede haber un Ranger sin Curricula.

Antes:



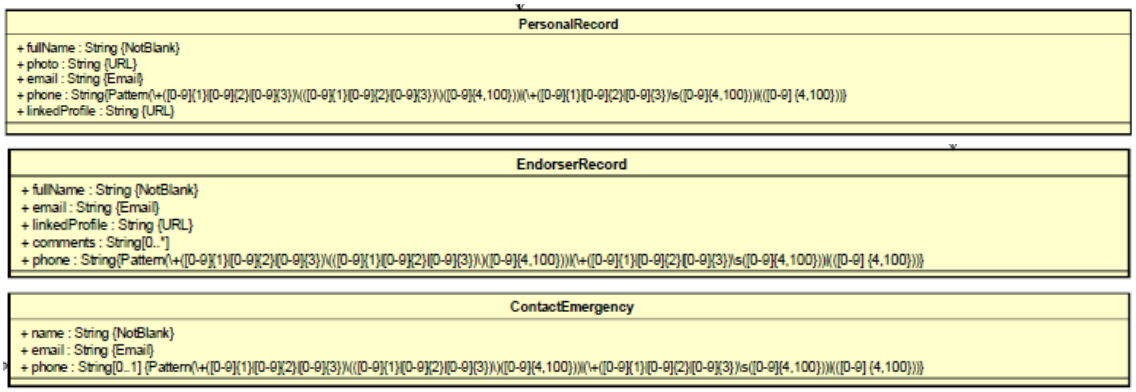
Después:



8. En las clases PersonalRecord, EndorserRecord y ContactEmergency hemos eliminado el patrón del atributo phone ya que aún no cumpliendo el patrón debe de guardarse en el sistema.

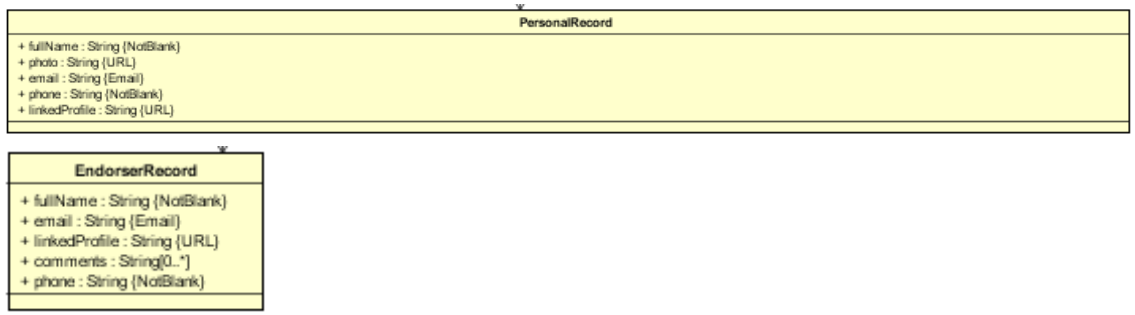
También se ha hecho un poco más pequeño dicho patrón para utilizarlo en un método en los servicios.

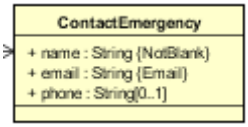
Antes:




```
(\+([0-9]{1}|[0-9]{2}|[0-9]{3})\(([0-9]{1}|[0-9]{2}|[0-9]{3})\)([0-9]{4,100}))|(\+([0-9]{1}|[0-9]{2}|[0-9]{3})\s([0-9]{4,100}))|(([0-9] {4,100}))
```

Después:






```
(\+\d{1,3})?\s?(\(\d{3}\))?\s?\d{4,100}
```

Cambios Realizados Java

1. En la clase AuditRecord hemos puesto @NotNull en el atributo isDraftMode debido a que no puede ser nulo.

Antes:

```
public boolean isDraftMode() {
    return this.draftMode;
}
```

Después:

```
@NotNull
public boolean isDraftMode() {
    return this.draftMode;
}
```

2. En la clase AuditRecord en el método getAttachments al devolvernos una colección hemos añadido @Valid por lo mismo que hemos explicado anteriormente en el punto 2.

Antes:

```
@NotNull
@ElementCollection
public Collection<String> getAttachments() {
    return this.attachments;
}
```

Después:

```
@NotNull
@ElementCollection
@Valid
public Collection<Attachment> getAttachments() {
    return this.attachments;
}
```

3. En la clase Story en el método getAttachments al devolvernos una colección hemos añadido @Vallid por lo mismo que hemos explicado anteriormente en el punto 2.

Antes:

```
@NotNull
@ElementCollection
public Collection<String> getAttachments() {
    return this.attachments;
}
```

Después:

```
@NotNull
@ElementCollection
@Valid
public Collection<Attachment> getAttachments() {
    return this.attachments;
}
```

4. En la clase Note hemos añadido el atributo body de tipo String para así poder ver el contenido de una nota.
   También hemos añadido sus correspondientes métodos getBody, el cual lleva la restricción @NotBlank, y setBody.

Antes:

```
private Date    createdMoment;
private int     remark;
private String  reply;
private Date    replyMoment;
```

Después:

```
private Date    createdMoment;
private int     remark;
private String  reply;
private Date    replyMoment;
private String  body;
public String getBody() {
    return this.body;
}

public void setBody(String body) {
    this.body = body;
}
```

5. En la clase Stage como el atributo totalPrice pasa a ser básico añadimos un setTotalPrice y el getTotalPrice lo modificamos.

Antes:

```
@Transient
public double getTotalPrice() {
    return this.price * 1.21;
}
```

Después:

```
public double getTotalPrice() {
    return this.totalPrice;
}


public void setTotalPrice(double totalPrice) {
    this.totalPrice = totalPrice;
}
```

6. En la clase CreditCard hemos cambiado el tipo de atributo de expirationMonth y expirationYear ya que vamos a restringirlo mediante un patrón.
También añadimos el patrón en los métodos getExpirationMonth y getExpirationYear así como el tipo que devuelven.
Modificacmos el tipo de atributo que se le pasa a los métodos setExpirationMonth y setExpirationYear por el tipo String.

Antes:

```
private String  holderName;
private String  brandName;
private String  number;
private int     expirationMonth;
private int     expirationYear;
private int     cvv;

public int getExpirationMonth() {
    return this.expirationMonth;
}

public int getExpirationYear() {
    return this.expirationYear;
}
public void setExpirationMonth(final int expirationMonth) {
    this.expirationMonth = expirationMonth;
public void setExpirationYear(final int expirationYear) {
    this.expirationYear = expirationYear;
}
```

Después:

```
private String  holderName;
private String  brandName;
private String  number;
private String  expirationMonth;
private String  expirationYear;
private int     cvv;
```

```
@Pattern(regexp = "^[0]{1}\\d{1}|[1]{1}[0-2]{1}$")
public String getExpirationMonth() {
    return this.expirationMonth;
}

@Pattern(regexp = "^\\d{2}$")
public String getExpirationYear() {
    return this.expirationYear;
}

public void setExpirationMonth(final String expirationMonth) {
    this.expirationMonth = expirationMonth;
}

public void setExpirationYear(final String expirationYear) {
    this.expirationYear = expirationYear;
}
```

7. En la clase Category al ponerla bidireccional añadimos el atributo fatherCategory así como sus correspondientes métodos getFatherCategory, setFatherCategory y sus restricciones. También eliminamos @Column del en el método getName del atributo name en la clase Trip.

Antes:

```
@NotBlank
@Column(unique = true)
public String getName() {
    return this.name;
}

private Collection<Category>    subCategories;
```

Después:

```
@NotBlank
public String getName() {
    return this.name;
private Collection<Category>    subCategories;
private Category                fatherCategory;
```

```
@Valid
@ManyToOne(optional = true)
public Category getFatherCategory() {
    return this.fatherCategory;
}

public void setFatherCategory(final Category fatherCategory) {
    this.fatherCategory = fatherCategory;
}
```

8. En la clase Curricula al cambiar la navegabilidad hemos añadido el atributo ranger con sus respectivos métodos getRanger y setRanger.

Antes:

```
private PersonalRecord                  personalRecord;
private Collection<EducationRecord>     educationRecords;
private Collection<ProfessionalRecord>  professionalRecords;
private Collection<EndorserRecord>      endorserRecords;
private Collection<MiscellaneousRecord> miscellaneousRecords;
```

Después:

```
private Ranger                          ranger;
private PersonalRecord                  personalRecord;
private Collection<EducationRecord>     educationRecords;
private Collection<ProfessionalRecord>  professionalRecords;
private Collection<EndorserRecord>      endorserRecords;
private Collection<MiscellaneousRecord> miscellaneousRecords;

@Valid
@OneToOne(optional = false)
public Ranger getRanger() {
    return this.ranger;
}

public void setRanger(Ranger ranger) {
    this.ranger = ranger;
}
```

9. En la clase Ranger al cambiar la navegabilidad hemos eliminado el atributo curricula y también sus respectivos métodos getCurricula y setCurricula.

Antes:

```
private Curricula           curricula;
private Collection<Trip>    trips;

@Valid
@OneToOne(optional = true)
public Curricula getCurricula() {
    return this.curricula;
}

public void setCurricula(final Curricula curricula) {
    this.curricula = curricula;
}
```

Después:

```
private Collection<Trip>    trips;
```

10. En la clase Trip hemos eliminado la restricción de @NotEmpty de getStages ya que un trip puede no tener ninguna stage referenciada.

Antes:

```
@NotNull
@NotEmpty
@OneToMany(cascade = CascadeType.ALL)
@Valid
public Collection<Stage> getStages() {
    return this.stages;
}
```

Después:

```
@NotNull
@OneToMany(cascade = CascadeType.ALL)
@Valid
public Collection<Stage> getStages() {
    return this.stages;
}
```

11. En la clase ConfigurationSystem hemos añadido una colección con las categorios por defecto que tiene el sistema con sus correspondientes métodos getDefaultCategories y setDefaultCategories;

Antes:

```
private double              VAT;
private String              banner;
private Collection<String>  spamWords;
private Collection<String>  welcomeMessages;
```

Después:

```
private double                  VAT;
private String                  banner;
private Collection<String>      spamWords;
private Collection<String>      welcomeMessages;
private Collection<Category>    defaultCategories;

@ElementCollection
@NotNull
@NotEmpty
public Collection<Category> getDefaultCategories() {
    return this.defaultCategories;
}

public void setDefaultCategories(final Collection<Category> defaultCategories) {
    this.defaultCategories = defaultCategories;
}
```