Universitat de Barcelona

Facultat de Matemàtiques i Enginyeria Informàtica

Pràctica 1

Grau en Enginyeria Informàtica

Curs 2022-2023 | *Cinquè Semestre*

Disseny de Software (DS)

Autor:
Mario Vilar
David Díez

Professor:
Leandro Zardaín

12 d'octubre de 2022

Presentació de la pràctica

TripUB: App per mòbil per planificar fàcilment viatges i rutes.

Universitat de Barcelona

Índex

I

Introducció

En verdad que, según pienso, el universo es joven, y es reciente en este mundo, y no hace mucho que tuvo principio. Por esto hoy todavía ciertas artes se perfeccionan, hoy todavía progresan.

Lucreci

En aquesta pràctica se'ns demanava desenvolupar una sèrie de testos per a un club d'activitats *TripUB*, una app per ordinador de sobretaula per poder planificar fàcilment viatges i rutes.

L'objectiu d'aquesta pràctica és, en essència, començar a entendre el paradigma de Programació Orientada a Testos per tal de poder modelar, mitjançant la metodologia AGILE, la interfície de l'usuari amb el conjunt de classes i informació que se'ns dona. En efecte, caldrà gestionar rutes, allotjaments i clients.

1. Inicialment l'aplicació TripUB disposa ja de rutes ja definides i la usuària prèviament enregistrada a l'aplicació pot accedir-hi, explorar-les i modificar-les lleugerament al seu gust, així com reservar els allotjaments que ofereix l'aplicació.
2. Només obrir l'app de TripUB, s'ha de crear un nou compte com a nova persona enregistrada a l'aplicació o bé s'entra directament al compte, sense necessitat de fer login, ja que el sistema ja té guardades les seves credencials en el mòbil.
3. Quan la usuària crea un compte a l'app TripUB s'utilitza l'adreça de correu electrònic per a identificar-la de forma inequívoca, i a més, es demana el seu DNI o NIE, nom, cognoms, any de naixement i la seva adreça completa, la qual inclou arrer, nombre, codi postal i població.
4. L'aplicació permet planificar tant viatges de rutes llargues que consten de diferents etapes on es pernocta, com rutes més curtes, o tracks, per a excursions o passejades. També es poden fer combinacions de rutes llargues amb excursions entremig. L'aplicació recomana un cert transport per a realitzar un tram (Cotxe, Bicicleta o A Peu), tot i que després, durant la planificació, es pot canviar de mitjà de transport que es voldrà fer servir. Per qualsevol mitjà de transport es vol tenir la velocitat mitja per fer una estimació del temps que es trigarà en fer algun recorregut.
5. Així, l'app TripUB proporciona rutes que comencen i acaben en certes localitats situades per les comarques de Catalunya. Les rutes poden estar formades per diferents tipus de trams (trams entre etapes o trams de track). Quan es tracta d'una etapa, l'aplicació també facilita allotjaments de diferents tipus (hotels, cases rurals, càmpings o refugis), que es diferencien pel seu grau de comoditat, el preu però també segons si ofereixen o no esmorzar, Mitja Pensió o Pensió Completa. Els hotels ofereixen els tres

tipus d'àpats, les cases rurals no ofereixen dinar, els refugis només ofereixen esmorzars i en el càmping no es dona cap tipus de servei. Aquesta informació es té en compte per a calcular el cost total de la ruta així com el preu per persona i nit. Pel cost total de la ruta també es té en compte el cost del transport, si és el cas. Per exemple, si s'utilitza el cotxe es té la informació dels litres que gasta el cotxe als 100 kms i el preu de la gasolina.

6. Cada ruta té un nom, un identificador únic i una descripció. A la ruta es guarda també el seu nivell de dificultat (Alt, Mitjà o Baix) i si és circular o no, com a informació addicional a la distància total, el nombre de dies i el seu cost estimat, per permetre filtres en les consultes. La distància total es calcula segons la distància entre trams.
7. Cada ruta té un nom, un identificador únic i una descripció. A la ruta es guarda també el seu nivell de dificultat (Alt, Mitjà o Baix) i si és circular o no, com a informació addicional a la distància total, el nombre de dies i el seu cost estimat, per permetre filtres en les consultes. La distància total es calcula segons la distància entre trams.
8. En el cas que la ruta impliqui etapes en les què es fa nit, quan es planifica, cal reservar un dels allotjaments proposats per l'app. En l'aplicació, també es permet fer grups de persones que pensen realitzar juntes la ruta.

L'app TripUB a dissenyar assumeix que la introducció de les dades de les rutes ja s'ha realitzat. Així, l'app es centrarà a gestionar l'accés a la informació disponible (diferents consultes que permetin triar una ruta), així com fer grups i sortir d'un grup, planificar la ruta, reservar allotjaments, si és necessari, i calcular els costos. A part en el perfil, es podrà consultar l'històric de rutes fetes o les guardades per més endavant.

En relació a les consultes, inicialment l'app només oferirà consultes per obtenir les rutes per comarca o per població. En les cerques es mostra un resum de la ruta i la usuària pot clicar per veure més informació, i en el cas que estigui interessada, la pot seleccionar per planificar la seva pròpia experiència.

> Ens hem assegurat de comentar bé tot el nostre codi, així que no ens entretindrem amb això.

Llista de codis font o listings

II

Cos del treball

2.1

Solució de codi

Com que hi ha moltes classes, i algunes d'aquestes són molt evidents donat el model de domini, aquí posarem les més importants; en efecte, hi ha una correspondència molt alta entre la quantitat de codi que hem editat en una classe i la importància d'aquesta. Evidentment es podran trobar totes als fitxers que es lliuraran de la pràctica.

```
12      private Map<String,Allotjament> allotjamentMap;
13      private Map<String,Transport> transportMap;
14      private Map<String,Tram> tramsMap;
15
16      public Controller() {
17          initXarxaPersones();
18          initRutesMap();
19          initAllotjamentsMap();
20          initTransportMap();
21          initTramsMap();
22          initComarquesMap();
23      }
24
25      public void initComarquesMap() {
26          comarcaMap = new HashMap<>();
27
28          //De la Cerdanya fins al Mar
29          afegirComarcaToRuta("Cerdanya", "De la Cerdanya fins al Mar");
30          afegirComarcaToRuta("Ripollès", "De la Cerdanya fins al Mar");
31          afegirComarcaToRuta("Garrotxa", "De la Cerdanya fins al Mar");
32          afegirComarcaToRuta("Alt Empordà", "De la Cerdanya fins al Mar");
33          //Terres de l'Ebre
34          afegirComarcaToRuta("Terra Alta", "Terres de l'Ebre");
35          afegirComarcaToRuta("Ribera d'Ebre", "Terres de l'Ebre");
36          afegirComarcaToRuta("Montsià", "Terres de l'Ebre");
37          afegirComarcaToRuta("Baix Ebre", "Terres de l'Ebre");
38          //La Costa Brava
39          afegirComarcaToRuta("La Selva", "La Costa Brava");
40          afegirComarcaToRuta("Baix Empordà", "La Costa Brava");
41          afegirComarcaToRuta("Alt Empordà", "La Costa Brava");
42          //Ribera del Ter
```

```
        afegirComarcaToRuta("Ripollès", "Ribera del Ter");
        afegirComarcaToRuta("Osona", "Ribera del Ter");
        afegirComarcaToRuta("La Selva", "Ribera del Ter");
        afegirComarcaToRuta("Gironès", "Ribera del Ter");
        afegirComarcaToRuta("Baix Empordà", "Ribera del Ter");
    }



    /* inicialitzem una llista de trams de diferents tipus */
    public void initTramsMap() {
        this.tramsMap = new HashMap<>();

        Tram tram1 = new Tram(340, 30, new Comarca("Cerdanya"), new Comarca("Cerdanya"), "Puigcerdà-Bor");
        Tram tram2 = new TramTrack(340, 30, new Comarca("Cerdanya"), new Comarca("Cerdanya"), 200,0,"Track Puigcerdà-Bor");
        Tram tram3 = new TramEtapa(340, 30, new Comarca("Cerdanya"), new Comarca("Cerdanya"), true,new Quantitat(30),"Etapa Puigcerdà-Bor");
        Tram tram4 = new TramEtapa(322, 42, new Comarca("Cerdanya"), new Comarca("Cerdanya"), true,new Quantitat(60),"Special Puigcerdà-Bor");
        Tram tram5 = new TramEtapa(322, 42, new Comarca("Cerdanya"), new Comarca("Ripollès"), true,new Quantitat(60),"Special2 Puigcerdà-Bor");

        tramsMap.put("Puigcerdà-Bor", tram1);
        tramsMap.put("Track Puigcerdà-Bor", tram2);
        tramsMap.put("Etapa Puigcerdà-Bor", tram3);
        tramsMap.put("Special Puigcerdà-Bor", tram4);
        tramsMap.put("Special2 Track Puigcerdà-Bor", tram5);

        /* afegim un tram per testejar la reserva */
        this.rutaMap.get("De la Cerdanya fins al Mar").addTram(tram4);
        this.rutaMap.get("De la Cerdanya fins al Mar").addTram(tram5);
        this.rutaMap.get("De la Cerdanya fins al Mar").addTram(tram1);
    }

    /* inicialitzem una llista de transports */
    public void initTransportMap() {
```

```
76          this.transportMap = new HashMap<>();
77          transportMap.put("Caminant",new APeu(5));
78          transportMap.put("Canondale 329V",new Bicicleta(25,"Canondale 329V"));
79          transportMap.put("Seat León",new Cotxe(120,"Seat León",true,
80                  4f,new Quantitat(1.90f)));
81          transportMap.put("Tesla Model S",new Cotxe(150,"Ford Fiesta",true,
82                  0f,new Quantitat(0)));
83      }
84
85      /* inicialitzem una llista d'allotjaments */
86      public void initAllotjamentsMap() {
87          this.allotjamentMap = new HashMap<>();
88
89          allotjamentMap.put("Bon dia", new Camping("Bon dia", new Comarca("Barcelonès"),
90                  300, new Quantitat(10)));
91          allotjamentMap.put("Camping Pepe", new Camping("Camping Pepe", new Comarca("La Selva"),
92                  150, new Quantitat(7)));
93
94          allotjamentMap.put("Ades Cansar", new Refugi("Ades Cansar", new Comarca("Cerdanya"),
95                  200,
96                  new Quantitat(15), new Quantitat(20)));
97          allotjamentMap.put("Catalina", new Refugi("Catalina", new Comarca("Ripollès"),
98                  190,
99                  new Quantitat(12), new Quantitat(25)));
100
101         allotjamentMap.put("Iber", new CasaRural("Iber", new Comarca("Gironès"), 200,
102                 new Quantitat(20),
103                 new Quantitat(10), new Quantitat(35)));
104         allotjamentMap.put("Homeomorfisme", new CasaRural("Homeomorfisme", new Comarca("Garraf"),
105                 350, new Quantitat(30),
106                 new Quantitat(15), new Quantitat(40)));
107
108         allotjamentMap.put("Four seasons", new Hotel("Four seasons", new Comarca("Barcelonès"),
```

```
                500,
                new Quantitat(150), new Quantitat(20), new Quantitat(50), new Quantitat(75)));
        allotjamentMap.put("Melià", new Hotel("Melià", new Comarca("Cerdanya"), 400,
                new Quantitat(100), new Quantitat(25), new Quantitat(60), new Quantitat(70)));
    }

    public void initXarxaPersones()  {
        List<Persona> listSocis = new ArrayList<>();
        listSocis.add(new Persona("ajaleo@gmail.com", "ajaleoPassw7"));
        listSocis.add( new Persona("dtomacal@yahoo.cat", "Qwertyft5"));
        listSocis.add(new Persona("heisenberg@gmail.com", "the1whoknocks"));
        listSocis.add(new Persona("rick@gmail.com", "wabalabadapdap22"));
        listSocis.add( new Persona("nietolopez10@gmail.com", "pekFD91m2a"));
        listSocis.add(new Persona("nancyarg10@yahoo.com", "contra10LOadc"));
        listSocis.add( new Persona("CapitaCC@gmail.com", "Alistar10"));
        listSocis.add( new Persona("nauin2@gmail.com", "kaynJGL20"));
        listSocis.add( new Persona("juancarlos999@gmail.com", "staIamsA12"));
        listSocis.add( new Persona("judit121@gmail.com", "Ordinador1"));

        xarxaPersones = new XarxaPersones(listSocis);
    }

    public void initRutesMap() {
        rutaMap = new HashMap<>();
        addRuta("De la Cerdanya fins al Mar", "29/09/2021", 5);
        addRuta( "Terres de l'Ebre", "04/10/2021", 2);
        addRuta( "La Costa Brava", "10/10/2021", 10);
        addRuta("Ribera del Ter", "11/10/2021", 4);
    }

    private void addRuta(String nom, String dataText, int numDies) {
        rutaMap.put(nom, new Ruta(nom, dataText, numDies));
    }
```

```
142
143     public List<Ruta> getRutaList(){
144         return new ArrayList<>(rutaMap.values());
145     }
146
147     public List<Tram> getTramsList(){
148         return new ArrayList<>(tramsMap.values());
149     }
150
151     public List<Transport> getTransportList(){
152         return new ArrayList<>(transportMap.values());
153     }
154
155     public boolean isPasswordSegur(String password) {
156         Pattern pattern = Pattern.compile("^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9]).{8,}$");
157         Matcher matcher = pattern.matcher(password);
158         return matcher.find();
159     }
160     public boolean isMail(String correu) {
161         Pattern pattern = Pattern.compile("^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
162                 + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$");
163         Matcher matcher = pattern.matcher(correu);
164         return matcher.find();
165     }
166
167     public Iterable<String> llistarCatalegRutesPerNom() {
168         SortedSet<String> excursionsDisponibles = new TreeSet<>();
169         if (getRutaList().isEmpty()) {
170             excursionsDisponibles.add("No hi ha excursions disponibles");
171         } else {
172             for (Ruta r : getRutaList()) {
173                 excursionsDisponibles.add(r.getNom());
174             }
```

```
        }
        return excursionsDisponibles;
    }

    public Iterable<String> llistarCatalegRutesPerDurada(){
        List<Ruta> sortedList = getRutaList();
        sortedList.sort(new Comparator<Ruta>() {
            public int compare(Ruta a1, Ruta a2) {
                return (Integer.compare(a1.getDurada(), a2.getDurada()));
            }
        });

        List<String> excursionsDisponibles = new ArrayList<>();
        for (Ruta r : sortedList) {
            excursionsDisponibles.add(r.getNom());
        }

        return excursionsDisponibles;
    }

    public String findPersona(String username) {
        Persona persona = xarxaPersones.find(username);
        if (persona!=null) return "Persona ja existent en el Sistema";
        else return "Persona desconeguda";
    }


    public String validatePassword(String b) {
        if (!isPasswordSegur(b)) {
            return "Contrassenya no prou segura";
        } else
            return "Contrassenya segura";
    }
```

```
208
209
210     public String validateUsername(String b) {
211         if (!isMail(b))
212             return "Correu en format incorrecte";
213         else
214             return "Correu en format correcte";
215     }
216
217     public String validateRegistrePersona(String username, String password) {
218         if (isMail(username) && isPasswordSegur(password)) {
219             Persona persona = xarxaPersones.find(username);
220             if (persona != null) {
221                 return "Persona duplicada";
222             } else return "Registre vàlid";
223         } else return "Format incorrecte";
224     }
225
226     public String loguejarPersona(String username, String password){
227         Persona persona = xarxaPersones.find(username);
228         if(persona == null){
229             return "Correu inexistent";
230         }
231         if(persona.getPwd().equals(password)){
232             return "Login correcte";
233         }else{
234             return "Contrassenya incorrecta";
235         }
236     }
237
238     public String recuperarContrassenya(String username){
239         Persona persona = xarxaPersones.find(username);
240         if(persona == null){
```

```
241             return "Correu inexistent";
242         }
243         return persona.getPwd();
244     }
245 
246 
247     public Comarca afegirComarca(String nomComarca){
248         Comarca comarca;
249         if(comarcaMap.containsKey(nomComarca)){
250             comarca = comarcaMap.get(nomComarca);
251         }else{
252             comarca = new Comarca(nomComarca);
253             comarcaMap.put(nomComarca, comarca);
254         }
255         return comarca;
256     }
257 
258     public void afegirComarcaToRuta(String nomComarca, String nomRuta){
259         Comarca comarca = afegirComarca(nomComarca);
260 
261         rutaMap.get(nomRuta).addComarca(comarca);
262 
263     }
264 
265     public Iterable<String> cercaRutesPerComarca(String nomComarca){
266         SortedSet<String> comarques = new TreeSet<>();
267 
268         if (comarcaMap.size() == 0){
269             comarques.add("No hi han comarques enregistrades");
270             return comarques;
271         }
272 
273         Comarca comarca = comarcaMap.get(nomComarca);
```

```
274         if (comarca == null)
275             comarques.add("Comarca no trobada en el sistema");
276 
277         else {
278             int ncount = 0;
279             for (Ruta ruta : rutaMap.values()){
280                 if (ruta.containsComarca(comarca)) {
281                     comarques.add(ruta.getNom());
282                     ncount++;
283                 }
284             }
285             if (ncount == 0) comarques.add("No hi han rutes en aquesta comarca");
286         }
287 
288         return comarques;
289     }
290 
291     private int comptarRutesComarca(Comarca comarca){
292         int count = 0;
293         for (Ruta ruta : rutaMap.values()){
294             if(ruta.containsComarca(comarca)) count++;
295         }
296         return count;
297     }
298 
299     /* afegim tramTrack, de manera totalment anàloga a com afegiríem un tram normal, sol que necessitem més dades */
300     /* afegir el tram a una ruta */
301     public String afegirTram(String nomRuta, String nomTram) {
302         /* Hem de buscar si el tram que volem afegir existeix a la llista de trams. */
303         Tram tram = null;
304         for (Tram iter : tramsMap.values())
305         {
306             /* Busquem si el nom del tram coincideix amb algun de la llista de trams */
```

```
307             if (iter.getId().equals(nomTram))
308             {
309                 /* En cas afirmatiu el guardem a una variable */
310                 tram = iter;
311             }
312         }
313 
314         boolean bool = false;
315 
316         if(rutaMap.isEmpty()) return "La llista de rutes està buida";
317         if(tram == null) return "Aquest tram no existeix";
318 
319         /* Iterem per totes les rutes */
320         for (Ruta ruta : rutaMap.values()){
321             /* ens quedem amb la ruta on vulguem afegir el tram */
322             if(ruta.getNom().equals(nomRuta)) {
323                 /* l'eliminem del mapa de rutes ja que després la tornarem a afegir actualitzada */
324                 ruta = this.rutaMap.remove(nomRuta);
325                 if (ruta == null) return "La ruta on estàs intentant afegir un tram no existeix";
326                 /* afegim el tram a la ruta */
327                 bool = ruta.addTram(tram);
328                 if(!bool) return "El tram que estàs intentant afegir ja es troba a la ruta";
329                 /* tornem a afegir la ruta al mapa de rutes */
330                 this.rutaMap.put(nomRuta,ruta);
331                 return "Hem afegit el tram " + nomTram + " correctament a la ruta " + nomRuta;
332             }
333         }
334         return "La ruta on estàs intentant afegir no està a la llista. Linca-la i torna a intentar-ho.";
335     }
336 
337     public String eliminarTram(String nomRuta, String nomTram)
338     {
339         /* Hem de buscar si el tram que volem eliminar existeix a la llista de trams. */
```

```
340         Tram tram = null;
341         for (Tram iter : tramsMap.values())
342         {
343             /* Busquem si el nom del tram coincideix amb algun de la llista de trams */
344             if (iter.getId().equals(nomTram))
345             {
346                 /* En cas afirmatiu el guardem a una variable */
347                 tram = iter;
348             }
349         }
350 
351         boolean bool = false;
352 
353         if(rutaMap.isEmpty()) return "La llista de rutes està buida";
354         if(tram == null) return "Aquest tram no existeix";
355 
356         /* Iterem per totes les rutes */
357         for (Ruta ruta : rutaMap.values()){
358             /* ens quedem amb la ruta on vulguem eliminar el tram */
359             if(ruta.getNom().equals(nomRuta)) {
360                 /* l'eliminem del mapa de rutes ja que després la tornarem a afegir actualitzada */
361                 ruta = this.rutaMap.remove(nomRuta);
362                 if (ruta == null) return "La ruta on estàs intentant eliminar un tram no existeix";
363                 /* eliminem el tram de la ruta */
364                 bool = ruta.deleteTram(tram);
365                 if(!bool) return "El tram que estàs intentant eliminar no es troba a la ruta";
366                 /* tornem a afegir la ruta al mapa de rutes */
367                 this.rutaMap.put(nomRuta,ruta);
368                 return "Hem eliminat el tram " + nomTram + " correctament de la ruta " + nomRuta;
369             }
370         }
371         return "La ruta on estàs intentant eliminar no està a la llista. Linca-la i torna a intentar-ho.";
372     }
```

```
373
374     /* aquest mètode mai no donarà errors en cas que li passem una ruta correcta */
375     public String modificarTransport(String nomRuta, String nomTram, String nomTransport) {
376         /* busquem a la llista de transports el que hem passat per paràmetre*/
377         Transport transport = transportMap.get(nomTransport);
378         if(transport == null) return "Transport no trobat.";
379         if(rutaMap.isEmpty()) return "Mapa de rutes buit";
380         /* busquem la ruta que estem fent */
381         for (Ruta ruta : rutaMap.values()){
382             /* ens quedem amb una ruta en concret */
383             if(ruta.getNom().equals(nomRuta)) {
384                 for(Tram iter : ruta.getTrams()) {
385                     /* escollim el tram del qual volem modificar un transport en concret */
386                     if(iter.getId().equals(nomTram)) {
387                         /* modifiquem el transport recomanat amb el mètode de l'element de la classe Tram */
388                         iter.setTransportRecomanat(transport);
389                         /* en cas que el canvi hagi sigut satisfactori, retornem true */
390                         return "El canvi ha sigut satisfactori, i hem modificat el transport recomanat";
391                     }
392                 }
393                 return "No hem pogut trobar el teu tram";
394             }
395         }
396         /* en cas que no s'hagi pogut trobar la ruta, o el tram, retornem fals per indicar que no hem pogut modificar
397          * res */
398         return "No hem pogut trobar la teva ruta. Pot ser que no l'hagis afegida correctament?";
399     }
400
401     public String resevarAllotjament(String nomRuta, String nomTram, String nomAllotjament)
402     {
403         String frase = "";
404
405         /* obtenim l'allotjament amb el nomAllotjament que passem per paràmetre */
```

```
406         Allotjament allotjament = allotjamentMap.get(nomAllotjament);
407
408         if(allotjament == null) return "Aquest allotjament no existeix";
409
410         if (rutaMap.isEmpty()) return "La llista de rutes està buida";
411
412         /* guardem en una variable la ruta que estem fent */
413         Ruta ruta = rutaMap.remove(nomRuta);
414
415         if(ruta == null) return "La ruta que ens has proporcionat no existeix";
416
417         /* guardem en una llista tots el trams de la ruta que estem fent */
418         List<Tram> tramsRuta = ruta.getTrams();
419
420         /* si no hi ha cap tram, obviament no podrem fer cap reserva */
421         if(tramsRuta.isEmpty()) return "La llista de trams està buida";
422
423         for (Tram iter : tramsRuta) {
424             /* ens quedem amb el tram que tingui el nom nomTram i que es pugui fer una reserva (tramEtapa) */
425             if (iter.getId().equals(nomTram)) {
426                 if (!(iter instanceof TramEtapa)) return "Aquest tram no permet fer reserves d'allotjaments.";
427                 /* fem downcasting i determinem si cal passar nit */
428                 if(((TramEtapa) iter).getEsFaNit()) {
429                     if(!(allotjament.getLloc().equals(iter.getFinal()))) return "Mala comarca";
430                     /* eliminem el tram de la ruta per tornar-lo a posar, a continuació, modificat */
431                     int index = ruta.getTrams().indexOf(iter);
432                     ruta.deleteTram(iter);
433                     /* MISSATGE IMPORTANT: això s'haurà de canviar de cara a les pràctiques següents
434                                            reservar l'allotjament a ruta.getDataCreacio() pot ser un error si la ruta
435                                            resulta ser de diversos dies. LA SOLUCIÓ, DE MOMENT, ÉS AQUESTA:
436                                            Posem la data de reserva amb un offset d'i dies, on i és la posició del
437                                            tram en l'ArrayList de trams d'aquella ruta.
438                     */
```

```
                    frase = ((TramEtapa) iter).
                            setAllotjament(allotjament,ruta.getDataCreacio().plusDays(index));
                    /* ens assegurem d'afegir el tram en la posició on estava */
                    ruta.addTram(iter,index);
                }
            }
        }
        rutaMap.put(nomRuta,ruta);
        /* si no hem modificat la frase, és a dir, si no s'ha fet la reserva, retornem un missatge d'error */

        if(frase.equals("")) {
            frase = "El tram que ens has donat, " + nomTram + ", no està dins d'aquesta ruta";
        }
        return frase;
    }

}
```

Listing 1: Classe *Controlador* del nostre projecte.

```
public class Ruta {
    private String nom;
    private int durada;
    private LocalDate dataCreacio;
    private Set<Comarca> comarques;
    private ArrayList<Tram> trams;
    private String dificultat;
    private String tipus;

    public Ruta(String titol, String dataText, int numDies) {
        this.nom = titol;
        setDurada(numDies);
```

```
        setDataCreacio(dataText);
        comarques = new HashSet<>();
        trams = new ArrayList<Tram>();
        setTipus("no s'ha definit tipus (circular o lineal) per a aquesta ruta");
        setDificultat("no s'ha definit una dificultat específica per a aquesta ruta");
    }

    public Ruta(String titol, int numDies, String dataText, List<Comarca> comarcaList, ArrayList<Tram> trams, String dificultat, String tipus) {
        this.nom = titol;
        setDurada(numDies);
        setDataCreacio(dataText);
        comarques = new HashSet<>(comarcaList);
        this.trams = new ArrayList<Tram>(trams);
        setTipus(tipus);
        setDificultat(dificultat);
    }

    private void setTipus(String tipus) {
        this.tipus = tipus;
    }

    public String getTipus() {
        return tipus;
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

```

```
55      public int getDurada(){
56          return this.durada;
57      }
58
59      public LocalDate getDataCreacio() { return dataCreacio;}
60
61      public void setDurada(int numdies){
62          this.durada = numdies;
63      }
64
65      public void setDataCreacio(String dataText){
66          DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
67          this.dataCreacio = LocalDate.parse(dataText, formatter);
68      }
69      public void addComarca(Comarca comarca){
70          comarques.add(comarca);
71      }
72
73      public boolean containsComarca(Comarca comarca){
74          return comarques.contains(comarca);
75      }
76
77      public List<Comarca> getComarques() { return new ArrayList<Comarca>(comarques);}
78
79      public String getDificultat() {
80          return dificultat;
81      }
82
83      public void setDificultat(String dificultat) {
84          this.dificultat = dificultat;
85      }
86
87      public ArrayList<Tram> getTrams() {
```

```
88          return trams;
89      }
90
91      public void setTrams(ArrayList<Tram> trams) {
92          this.trams = trams;
93      }
94
95      /* no té molt de misteri, afegim el tram si no hi és */
96      public boolean addTram(Tram tram) {
97          if(!trams.contains(tram))
98          {
99              trams.add(tram);
100             return true;
101         }
102         return false;
103     }
104
105     /* el mateix mètode que abans, però afegint el tram en un índex concret */
106     public boolean addTram(Tram tram, int index) {
107         if(!trams.contains(tram) && index >= 0)
108         {
109             trams.add(index,tram);
110             return true;
111         }
112         return false;
113     }
114
115     /* eliminem el tram si es troba a la llista de trams */
116     public boolean deleteTram(Tram tram)
117     {
118         if(trams.contains(tram))
119         {
120             trams.remove(tram);
```

```
            return true;
        }
        return false;
    }
}
```

Listing 2: Classe *Ruta* del nostre projecte.

```
package ub.edu.model;

import javax.swing.*;
import java.util.*;

public class Tram
{
    private int tempsEstimat; // en minuts
    private float distancia;
    private Transport transportRecomanat;
    private String id;

    private Comarca inici_etapa;
    private Comarca final_etapa;

    public Tram(int temps, float dist, Comarca inici_etapa, Comarca final_etapa, String id)
    {
        setTempsEstimat(temps);
        setDistancia(dist);
        setId(id);
        setTransportRecomanat(null);
        setInici(inici_etapa);
        setFinal(final_etapa);
    }
```

```
25
26      public void setTempsEstimat(int temps)
27      {
28          this.tempsEstimat = temps;
29      }
30
31      public void setDistancia(float dist)
32      {
33          this.distancia = dist;
34      }
35
36      public int getTempsEstimat()
37      {
38          return this.tempsEstimat;
39      }
40
41      public float getDistancia()
42      {
43          return this.distancia;
44      }
45
46      public Transport getTransportRecomanat() {
47          return transportRecomanat;
48      }
49
50      public void setTransportRecomanat(Transport trans) {
51          /* si el transport que se'ns passa per paràmetre és nul, en creem un nosaltres en funció de la distància */
52          if (trans == null) {
53              float dist = this.distancia;
54              Transport new_trans;
55              if (dist <= 5)
56              {
57                  new_trans = new APeu(5);
```

```
            }
            else if(dist > 5 && dist <= 50)
            {
                new_trans = new Bicicleta(20,"Cannondale");
            }
            else
            {
                new_trans = new Cotxe(80, "Seat León",true,5,new Quantitat(2));
            }
            trans = new_trans;
        }
        this.transportRecomanat = trans;
    }

    public String getId() {
        return id;
    }

    private void setId(String id) {
        this.id = id;
    }

    public Comarca getInici()
    {
        return this.inici_etapa;
    }

    public void setInici(Comarca inici_etapa)
    {
        this.inici_etapa = inici_etapa;
    }

    public Comarca getFinal()
```

```
91      {
92          return this.final_etapa;
93      }
94
95      public void setFinal(Comarca final_etapa)
96      {
97          this.final_etapa = final_etapa;
98      }
99
100 }
101
102 /*
103         Random rand_num = new Random();
104         int random;
105         int upper_bound = 10;
106         String pass = "";
107
108         for(int i = 0; i < upper_bound; i++)
109         {
110             random = rand_num.nextInt(upper_bound);
111             pass += Integer.toString(random);
112         }
113  */
```

Listing 3: Classe *Tram* del nostre projecte.

```
1  package ub.edu.model;
2
3  import java.time.LocalDate;
4
5  public class TramEtapa extends Tram
6  {
```

```
    private boolean esFaNit;

    private Quantitat cost;

    private Etapa etapa;

    public TramEtapa(int temps, float dist, Comarca inici_etapa, Comarca final_etapa, boolean esFaNit, Quantitat cost, String id)
    {
        super(temps, dist, inici_etapa, final_etapa, id);

        setEsFaNit(esFaNit);
        setCost(cost);

        /* de moment l'incialitzarem a null donat que no la necessitem en la nostra implementació */
        setEtapa(null);
    }

    public void setCost(Quantitat cost)
    {
        this.cost = cost;
    }

    public void setEsFaNit(boolean nit)
    {
        this.esFaNit = nit;
    }

    public Quantitat getCost()
    {
        return this.cost;
    }

    public boolean getEsFaNit()
```

```
    {
        return this.esFaNit;
    }

    /* necessitem l'allotjament i la data en què volem fer la reserva, per poder veure si l'allotjament està ple en
     * aquelles dates o no */
    public String setAllotjament(Allotjament allotjament, LocalDate date) {
        String comarcaDestinacio = this.getFinal().getNom();
        /* de moment, a falta d'una millor gestió de les dates, no mirem si està ple (!allotjament.isFull()) */
        /* comprovem que la comarca de l'allotjament i la comarca del final del tram coincideixen */
        if(comarcaDestinacio.equals(allotjament.getLloc().getNom()))
        {
            /* afegim la reserva via la classe allotjament */
            allotjament.addReserva(date);
            /* retornem true per denotar que hem afegit la reserva correctament a l'allotjament */
            return "Hem afegit la teva reserva correctament";
        }
        /* si la inserció no s'ha pogut fer, retornem false */
        return "La comarca de l'allotjament no coincideix amb la de final de tram";
    }

    public void setEtapa(Etapa etapa) {
        this.etapa = etapa;
    }
}
```

Listing 4: Classe *TramEtapa* del nostre projecte.

```
package ub.edu.model;

public class Transport
{
```

```
    private float velocitatEstimada;

    public Transport(float velocitat)
    {
        setVelocitatEstimada(velocitat);
    }

    public void setVelocitatEstimada(float velocitatEstimada)
    {
        this.velocitatEstimada = velocitatEstimada;
    }

    public float getVelocitatEstimada()
    {
        return this.velocitatEstimada;
    }
}
```

Listing 5: Classe *Transport* del nostre projecte.

```
package ub.edu.model;

import org.jetbrains.annotations.NotNull;
import java.math.*;

public class Quantitat {
    private float valor;
    public Quantitat(float valor)
    {
        setValor(valor);
    }

```

```
13      public void setValor(float valor)
14      {
15          this.valor = valor;
16      }
17
18      public float getValor()
19      {
20          return this.valor;
21      }
22
23      public boolean equals(Quantitat other)
24      {
25          return Math.abs(this.getValor() - other.getValor()) < Math.pow(10,-5);
26      }
27
28  }
```

Listing 6: Classe *Quantitat* del nostre projecte.

2.2

Preguntes de la pràctica

Donat el DCU, les primeres Històries d'Usuari i el Model de Domini de l'aplicació TripUB cal que el vostre equip de desenvolupament inclogui les següents funcionalitats.

Exercici 2.2.1. *Planificar un viatge personalitzat a partir de les rutes ja definides. A partir d'una ruta, l'usuari pot esborrar trams, canviar els mitjans de transport associats al tram, si és possible.*

Resolució. En essència, hem de realitzar els tests que se'ns demana; això és, crear els nostres propis viatges. Després, per tal que els testos passin, serà necessari implementar els nostres propis mètodes. De fora cap a dins: hem de poder personalitzar les rutes internament per afegir o eliminar trams, dins dels trams els mitjans de transport es podran modificar.

1. **Esborrar trams**: per esborrar trams creem un nou test `EsborrarTrams`. Per llistar les rutes d'on vulguem eliminar els trams tenim el test `llistaRutes`.
   - Cal que la ruta que li passem per paràmetre s'hagi definit prèviament.
   - Cal que els trams que vulguem eliminar estiguin inclosos a la ruta que li passem per paràmetre.
   - Cal que el tram i la ruta estiguin ben definits (no siguin nuls, ni res per l'estil).
2. **Afegir trams**: per afegir trams creem un nou test `AfegirTrams`. ídem *Esborrar trams*, excepte que cal que *no* estiguin inclosos.
3. **Modificar transport recomanat**: per modificar un transport recomanat creem un nou test. Per modificar el transport recomanat s'han de passar per paràmetre un objecte `Ruta`, un objecte `Tram` i un `Transport`; volem fixar un nou transport d'un tram concret d'una ruta determinada.
   - Cal que la ruta i el tram que passem per paràmetre s'hagin definit prèviament. Si la ruta no es troba a la llista de rutes, si el tram no es troba a la llista de trams de la ruta, retornem un `String` d'error.
   - Cal que el tram, la ruta i el transport estiguin ben definits (no siguin nuls, ni res per l'estil).
4. **Reservar allotjament**: per reservar un nou allotjamemnt creem un nou test `ReservarAllotjament`. El comentarem extensament més endavant així que no afegirem res més ara.
   - Cal que la ruta que li passem per paràmetre s'hagi definit prèviament.
   - Cal que el tram existeixi dins la ruta.
   - Cal que el tram sigui de tipus `TramEtapa` i que s'hagi de passar la nit.
   - Cal que l'allotjament es trobi a la comarca de destí.
   - Cal que el tram i la ruta estiguin ben definits.

■

Exercici 2.2.2. *Reservar allotjament a les rutes planificades. Només a les etapes de les rutes planificades es poden reservar allotjaments.*

Observació 2.2.3. Considera que des del perfil de l'usuari es podran llistar les rutes planificades i les llistes de reserva realitzades per l'usuari llistant la data d'inici i final de la reserva.

Resolució. Aquí ens hem trobat amb un problema força important: el temps. No és un actor en la nostra aplicació (de moment no deixa de ser una pràctica d'universitat), sinó que el propi usuari s'encarregarà d'especificar les dates quan les necessiti. En aquest sentit, hem de reservar allotjament solament en aquells `TramEtapa` on s'hagi de fer nit. Això vol dir que, altrament, no deixarà fer la reserva.

Dins la classe Allotjament hem creat una llista de dates per a denotar la quantitat de reserves d'un dia determinat. Dit això, l'única data que coneixem en el nostre Model de Domini és la del començament de la ruta i hem de reservar el nombre d'allotjaments necessaris en funció del nombre d'etapes `TramEtapa` amb nit que hi hagi (de fet, si la ruta és de 8 dies, hi hauria d'haver 7 `TramsEtapa` amb nit com a mínim). El problema és com assignem les dates de reserva coneixent únicament la data d'inici de la ruta. Es poden assignar de manera aleatòria? Com ho gestionem si hem de treballar amb un `String` (revisar el mètode `setDataCreacio()` de la classe `Ruta`)? De moment això no hauria d'influir en el desenvolupament dels testos. Esperem que se'ns donin les eines suficients per a arreglar aquesta falla en pràctiques posteriors. ■

Exercici 2.2.4. *Donades aquestes noves funcionalitats, com es relacionen amb els casos d'ús ja disponibles? Quines noves històries d'usuari sorgeixen? Quin canvis cal aplicar al Model de Domini de la primera solució?*

Resolució. Els casos d'ús ja disponibles ens permeten establir uns nous casos d'ús que, al seu torn, ens donaran noves històries d'usuari.

1. Planificar ruta: es divideix en *Reservar allotjament* i *Modificar ruta*. Per tant, els subcasos d'ús que generem determinen aquest cas d'ús.
2. Reservar allotjament: *com a persona enregistrada vull resevar un allotjament per a poder passar la nit*:
   - Allotjament no disponible: En cas que l'allotjament estigui ple o no existeixi quan l'usuari vol reservar un allotjament el sistema mostrarà el missatge "Aquest allotjament està ple".
   - Allotjament disponible: En cas que hi hagi espai per a dormir quan l'usuari vol reservar un allotjament el sistema mostrarà el missatge "Reserva feta correctamentï l'apuntarà a la llista de reserves.
3. Modificar ruta (Afegir Tram): Com a persona enregistrada vull modificar la ruta per a fer una excursió que m'agradi.

- La ruta buscada no existeix: En cas que la ruta no existeixi en la base de dades quan l'usuari vol afegir un tram el sistema mostrarà el missatge "Ruta no existent".
- Si el tram no existeix en la base de dades: En cas que el tram no existeixi en la base de dades quan l'usuari vol afegir un tram el sistema mostrarà el missatge "Tram no existent".
- Si el tram ja està a la ruta: En cas que la ruta que intento modificar tingui el tram afegit quan l'usuari vol afegir un tram el sistema mostrarà el missatge "Aquest tram ja ha estat afegit"
- Si tot va bé: En cas que tot sigui exitós quan l'usuari vol afegir un tram el sistema mostrarà el missatge "S'ha afegir el tram correctament".

4. Modificar ruta (Afegir Tram): Com a persona enregistrada vull modificar la ruta per a fer una excursió que m'agradi.
    - La ruta buscada no existeix: En cas que la ruta no existeixi en la base de dades quan l'usuari vol afegir un tram el sistema mostrarà el missatge "Ruta no existent".
    - Si el tram no existeix en la base de dades: En cas que el tram no existeixi en la base de dades quan l'usuari vol afegir un tram el sistema mostrarà el missatge "Tram no existent".
    - Si el tram no està a la ruta: En cas que la ruta que intento modificar no tingui el tram afegit quan l'usuari vol afegir un tram el sistema mostrarà el missatge "Aquest tram no ha estat afegit"
    - Si tot va bé: En cas que tot sigui exitós quan l'usuari vol afegir un tram el sistema mostrarà el missatge "S'ha afegir el tram correctament".
5. Modificar transport: Vull modificar el transport per a poder adaptar-me a les meves capacitats.
    - La ruta buscada no existeix: En cas que la ruta no existeixi en la base de dades quan l'usuari vol afegir un transport el sistema mostrarà el missatge "Ruta no existent".
    - Si el tram no existeix en la base de dades: En cas que el tram no existeixi en la base de dades quan l'usuari vol afegir un transport el sistema mostrarà el missatge "Tram no existent".
    - Si el transport no existeix: En cas que no existeixi el transport quan l'usuari vol afegir un transport a un tram el sistema mostrarà el missatge "Aquest transport no existeix"
    - Si tot va bé: En cas que tot sigui exitós quan l'usuari vol afegir un transport recomanat el sistema mostrarà el missatge "S'ha afegir el transport recomanat correctament".
6. Sortir del grup: Vull sortir del grup d'excursions per a no formar part del grup i no realitzar l'excursió. *No l'implementarem, ja que cap dels exercicis ho demana a posteriori*.
    - No pertanys a cap grup: en cas que l'usuari encara no pertanyi a cap grup quan l'usuari seleccioni sortir del grup el sistema mostrarà el missatge "No pertanys a cap grup".
    - Pertanys al grup: En cas que l'usuari si que pertanyi a algun grup quan l'usuari seleccioni sortir

del grup el sistema mostrarà el missatge "Ja has sortit del grupï el farà fora.

Podeu trobar les històries d'usuari completes amb els seus testos d'acceptació a l'Excel.

A la següent secció mostrem el nostre model de domini, perquè es pugui consultar quins canvis s'han d'aplicar al Model de Domini de la primera solució directament des d'aquesta memòria, sense entrar al projecte. Bàsicament cal tenir en compte les noves funcionalitats que se'ns demana, la dependència que té aquesta aplicació de la classe `Ruta` i tenir en compte la classe `Quantitat`. A la vegada, no ens podem oblidar certs atributs que s'han deixat fora del model de domini existent (fet a consciència o no, no hi són i s'han de posar). ■

2.3 Diagrames

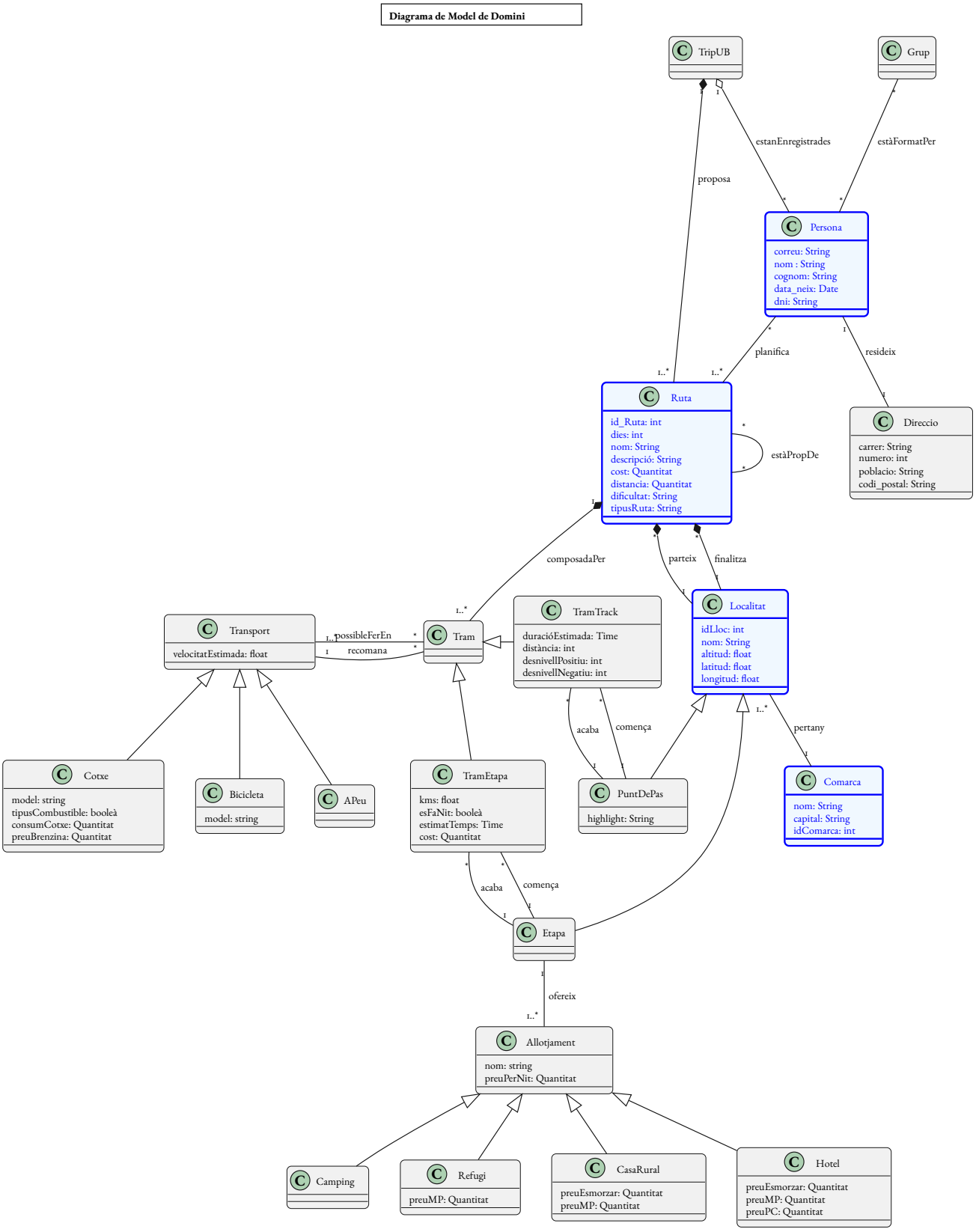

Figura 2.1: Diagrama del nostre model de domini resultant.

En efecte, cal afegir els atributs de la classe `Cotxe` perquè es pugui parametritzar el preu de la benzina i altres. A partir d'aquí, es podrà veure a les classes que hem afegit a les seccions anteriors totes les relacions que cal establir entre les classes, atributs que ens ha calgut afegir per tal que el nostre programa pugui funcionar correctament.

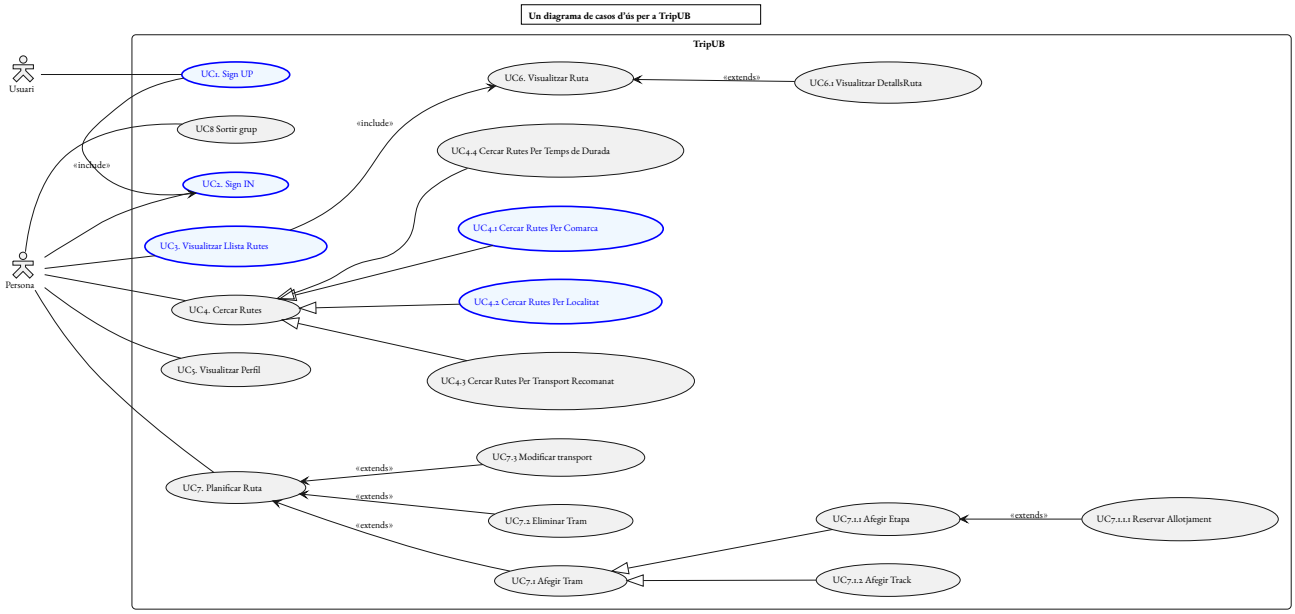

Figura 2.2: Diagrama de casos d'ús del nostre projecte.

Pel que fa al diagrama de casos d'ús, hem de poder sortir del grup (es pot veure a l'enunciat de la pràctica com s'ha de contemplar), però no ho implementarem. Ara, cal planificar la ruta i, dins d'aquest cas d'ús, hem de poder (que no, estem obligats) a afegir un tram o bé eliminar-ne un, així com modificar el transport recomanat. En aquest sentit, podem distingir dos tipus de trams a afegir, i, a més, quan afegim un tram de tipus Etapa tindrem la possibilitat de reservar allotjament en cas que estiguem forçats a fer nit durant l'etapa. No entrarem en detall en la justificació, ja que són simples directius de l'enunciat.

III

Conclusions

En aquesta pràctica hem pogut consolidar el nostre coneixement sobre en la programació orientada a testos, els models de domini, les històries d'usuari i els casos d'ús. En particular, en el llenguatge `Java`.

Ens hem hagut de familiaritzar molt bé amb molts conceptes d'aquest paradigma en el transcurs de les darreres tres setmanes, ja que els conceptes de *model de domini* i *històries d'usuari*, entre d'altres, ens venien de nou. La part que més ens ha costat, segurament, ha sigut la implementació de les reserves, ja que l'ús de dates ens ho ha fet impossible a l'hora de planificar un allotjament en un dia concret. També hem hagut de parar especial atenció amb l'entrada de dades: els paràmetres han de ser d'un cert tipus; en particular, comentar el cas de `Quantitat`. Solament els preus i les divises s'han de posar com a objectes d'aquesta classe. Nosaltres no ho teníem entès així, i de cara al final de la pràctica ho hem hagut de canviar tot un altre cop. Hem tingut, per últim, alguns dubtes de com orientar els nostres testos, però sembla que al final ens n'hem sortit bé. Per altra banda, si hem de destacar una cosa que hem fet bé és intentar adaptar-nos a les bones pràctiques de programació des del principi: hem usat construccions genèriques sempre que hem pogut, hem adaptat els noms de les variables a les convencions de `Java` i hem intentat estalviar línies de codi innecessàries, així com fer un control d'errors robust i descentralitzat en les diferents classes, per fer-ho més heurístic.

En definitiva, hem assolit l'objectiu del programa que se'ns proposava a l'inici de la pràctica.

Bibliografia

[Bec03] Kent Beck. *Test-driven development : by example*. eng. The Addison-Wesley signature series. Boston [etc: Addison-Wesley, 2003. isbn: 0321146530.

[Lew07] John Lewis. *Java software solutions : foundations of program design*. eng. 5th ed. Boston: Pearson Addison-Wesley, 2007. isbn: 9780321409492.

[Lar16] Craig Larman. *Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development*. eng. Third edition. Chennai: Pearson, 2016. isbn: 9789332553941.