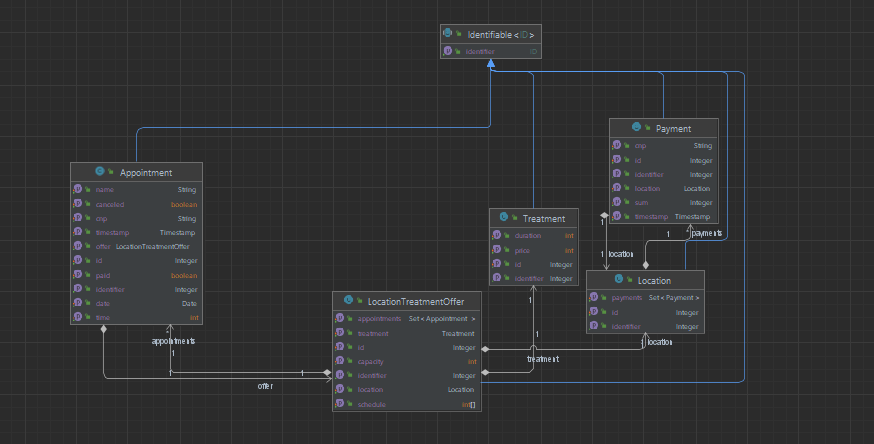
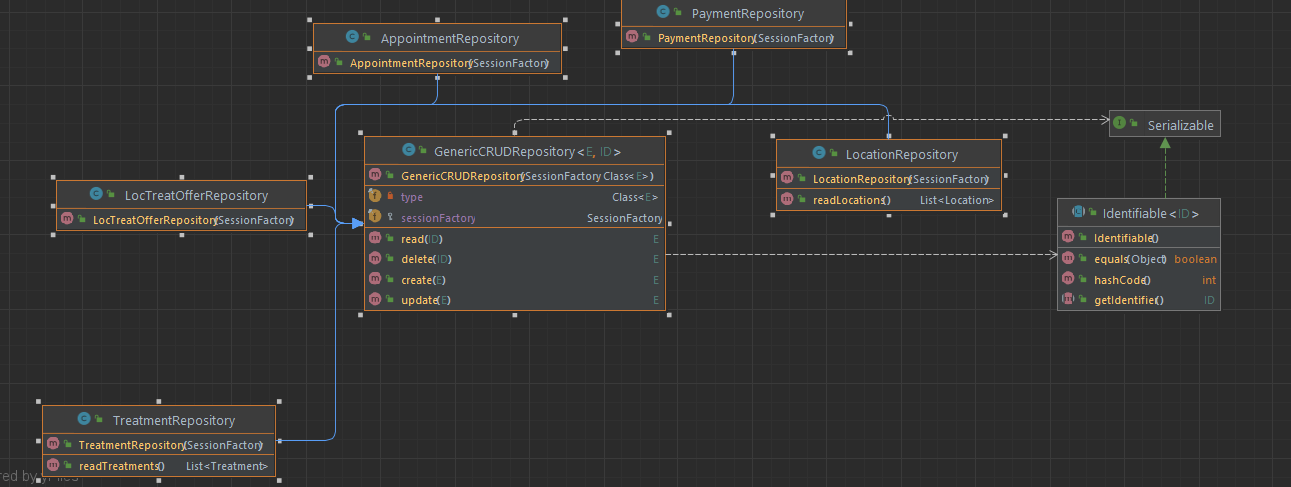
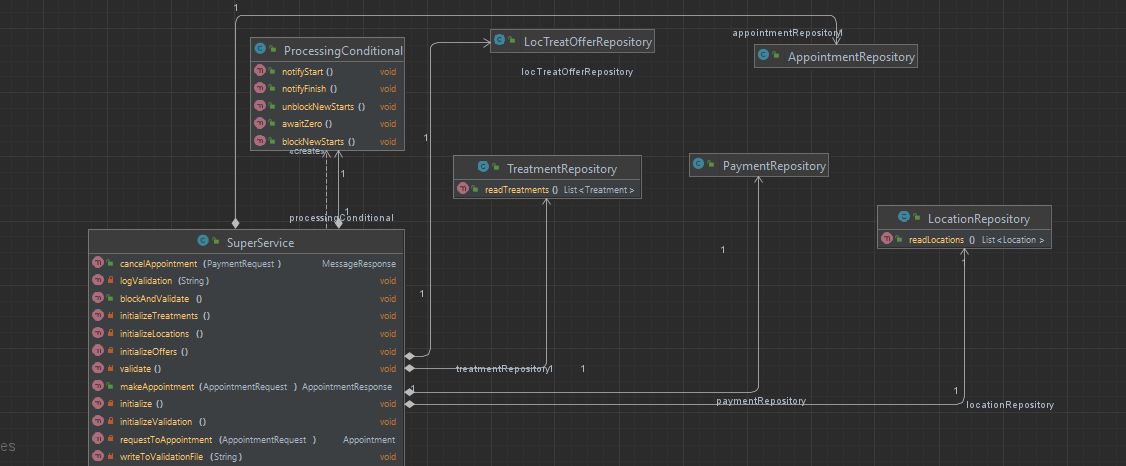
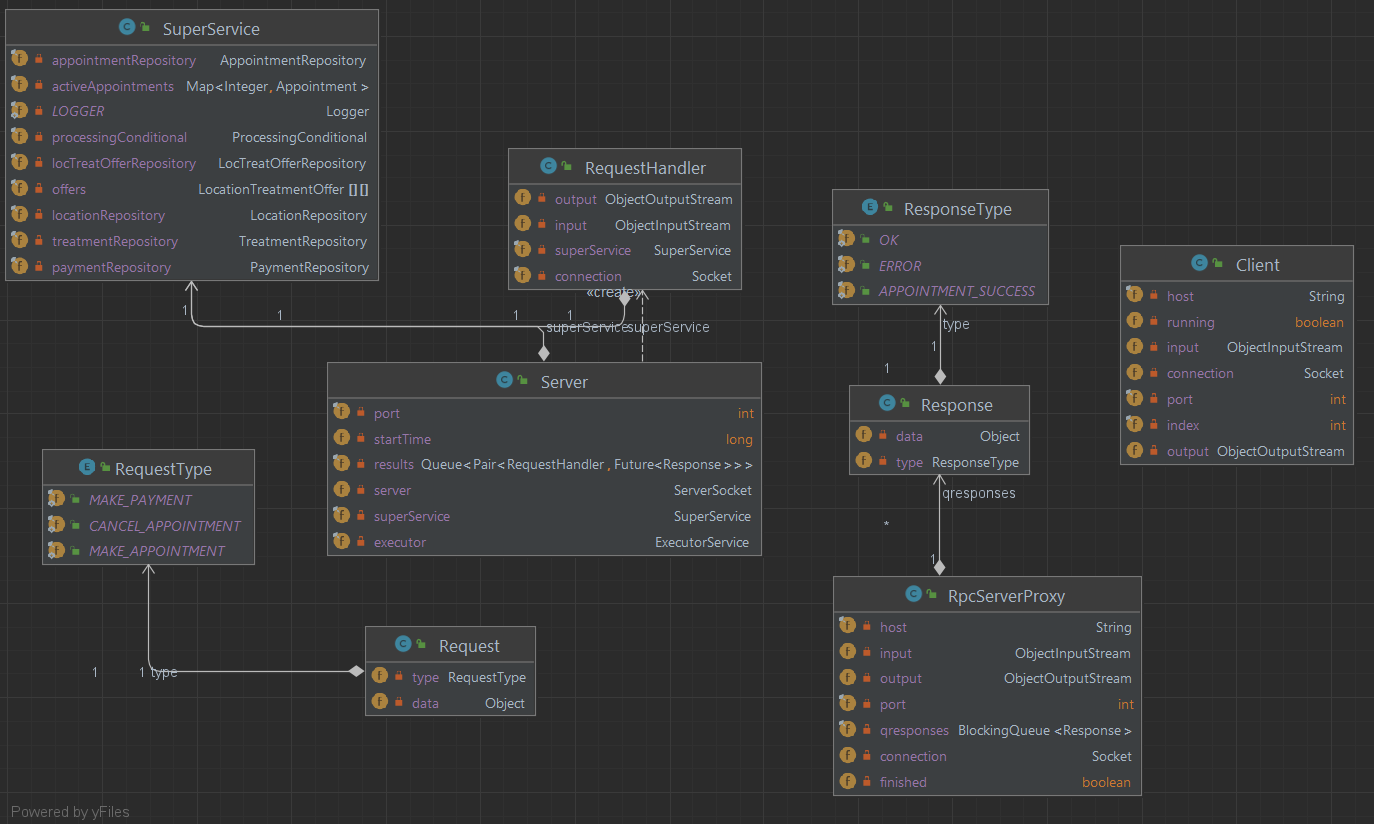
Documentatie PPD P1

* Info:
  + Device 1:
    - OS: Windows 10
    - Processor: 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz 2.30 GHz
    - Cores: 8
    - Logical Processors: 16
    - RAM: 16.0 GB (15.7 GB usable)
  + Device 2:
    - OS: Windows 10
    - Processor: AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz
    - Cores: 8
    - Logical Processors: 16
    - RAM: 16.0 GB (15.4 GB usable)
* Idee:
  + Model:
    - Locatie:
      * Este reprezentata de un simplu id si ea va tine cont de toate platile ce se realizeaza la ea
    - Plata:
      * Contine detaliile celui ce face plata si valoarea transferata
    - Tratament:
      * Este reprezentat de un id, un pret si o durata de realizare
    - Oferta:
      * Pentru fiecare locatie si tratament ce se poate realiza la locatia respectiva exista o oferta
      * Este descrisa de o capacitate
      * Pentru o oferta se pot realiza mai multe rezervari
    - Rezervare:
      * Contine detaliile celui ce face programarea
  + Generalitati sincronizare:
    - Model:
      * Locatia este folosita readonly, deci nu trebuie sincronizata
      * O plata o data realizeaza nu se mai modifica, deci nu trebuie sincronizata (read only)
      * Tratamentul este readonly, deci nu trebuie sincronizat
      * Oferta este si scrisa si citita, asa ca este sincronizata utilizand un lock – toate accesarile sunt gestionate din exterior folosind lock-ul
      * Rezervarea se modifica pe parcurs, asa ca este sincronizata utilizand un lock – toate accesarile sunt gestionare din exterior folosind lock-ul
    - Validare:
      * Pentru a realiza validarea, trebuie sa fim siguri ca nimeni nu realizeaza modificari asupra rezervarilor si platilor
      * Pentru asta, am realizat un mecanism de gestionare a accesului la rezervari/plati (R/P)
      * Atunci cand cineva doreste sa acceseze R/P, va face cerere la mecanismul de gestionare si va astepta permisiunea de la acesta
      * Atunci cand termina accesul, va anunta mecanismul ca a terminat accesul
      * Mecanismul de access contorizeaza cate procese sunt active
      * Atunci cand se doreste validarea, se anunta mecanismul de validare sa blocheze accesul si se asteapta sa nu mai fie procese active – cand se termina validarea, se anunta mecanismul de valiare ca poate permite din nou accesul
    - Comunicare:
      * Thread-urile sunt gestionate utilizand ExecutorService
      * Thread-ul prinicipal accepta conexiuni de la clienti si pentru fiecare creeaza un future ce returneaza raspunsul la cerere ce-l scrie intr-o coada de asteptare
      * Un thread citeste din coada de asteptare futures si le trimite clientilor raspunsurile generate
      * Un thread realizeaza periodic validarea
      * Fiecare future este creat dintr-un obiect special ce handle-uieste cererea
    - General:
      * Blocarile s-au incercat sa se faca pe blocuri cat mai mici, deci granularitatea este una cat mai mare
* Specificatii:
  + Specifcatiile sunt puse pe codul sursa, la fiecare metoda, impreuna cu detaliile extra de implementare
* Java:
  + Diagrame :









* + Rezultate:
    - Atat validarea la interval de 5, cat si cea de 10 secunde nu dau rateuri, dand mereu stari valide
    - Varianta pe 10 secunde reuseste sa raspunda bineinteles la un numar mai mare de request-uri in intervalul de 3 minute
* Overall Conclusions:
  + Implementarea curenta pare sa fie mai eficienta decat una secventiala
  + Validarea la 10 secunde este ideala pentru ca este indeajuns de deasa pentru a asigura integritatea aplicatiei, cat si indeajuns de rara pentru a nu afecta prea mult performanta