TicketApp jegyfoglaló rendszer

Dokumentáció

|  |  |  |
| --- | --- | --- |
| Applikáció: | TicketApp | |
| Készítő: | Simon Ádám | |
| Dokumentáció kezdete: | 2020.03.25. | Version: 1.0 |
| Kezelés és használat módja: | belső és korlátozott külső | |
| Konzultációs partner: | Simon Ádám, 06 20 386 1032 (adam.simon9303@gmail.com) | |
| A dokumentáció és az ez alapján készült applikáció egy felvételi feladat részét képezi. | | |

**Tartalomjegyzék**

[I. Áttekintés 2](#_Toc37095783)

[1. Változtatási történet 2](#_Toc37095784)

[2. Rövid leírás 2](#_Toc37095785)

[II. Diagramok 3](#_Toc37095786)

[1. Adatmodellek és entitások 3](#_Toc37095787)

[2. Use-case diagram 4](#_Toc37095788)

[3. Activity diagram 5](#_Toc37095789)

[III. Modulok 7](#_Toc37095790)

[1. Appl modul 7](#_Toc37095791)

[AuthFilter 7](#_Toc37095792)

[AuthProvider 7](#_Toc37095793)

[SecurityConfig 8](#_Toc37095794)

[Data.sql 8](#_Toc37095795)

[Application.properties 8](#_Toc37095796)

[2. Commons modul 8](#_Toc37095797)

[3. Api modul 8](#_Toc37095798)

[ApiController 8](#_Toc37095799)

[ApiService 9](#_Toc37095800)

[4. Core Modul 9](#_Toc37095801)

[User, UserBankCard, UserRepository, UserBankCardRepository 9](#_Toc37095802)

[IncomingRequestService 10](#_Toc37095803)

[HelperService 11](#_Toc37095804)

[DataBaseHandlerService 11](#_Toc37095805)

[OutGoingService 12](#_Toc37095806)

[Core.properties 12](#_Toc37095807)

[5. Ticket Modul 12](#_Toc37095808)

[UserToPartner, UserRepositoryToPartner 12](#_Toc37095809)

[TicketDatabaseCallerService 13](#_Toc37095810)

[TicketBuyingService 13](#_Toc37095811)

[PartnerCallerService 14](#_Toc37095812)

[6. Partner modul 15](#_Toc37095813)

[TicketModuleUser, TicketModuleUserRepository 15](#_Toc37095814)

[PartnerDatabaseCallerService 15](#_Toc37095815)

[PartnerController 15](#_Toc37095816)

[PartnerService 16](#_Toc37095817)

[IV. Függelék 17](#_Toc37095818)

[1. További tervezett munkálatok 17](#_Toc37095819)

# Áttekintés

## Változtatási történet

| **Verzió** | **Dátum** | **Készítő** | **Leírás** |
| --- | --- | --- | --- |
| 1.0 | 2020.03.25. | Simon Ádám | A dokumentáció elkészítése az applikáció magjához. |
| 1.1 | Tervezett | Simon Ádám | TERVEZETT:  Sikeres foglaláskor vonja le az egyenlegből az összeget.  Sikeres vásárláskor a Partner JSON-ben írja át az aktuális EventDetails objektum megfelelő Seat objektumának „reserved” mezőjét. |

## Rövid leírás

A TicketApp applikáció, mely ezen dokumentum, tárgyát képezi az OTP Mobil Kft-hez készített felvételi feladat részét képezi. Az applikáció egy jegyfoglalási rendszer vázát adja, melyben lehetőség van az események lekérdezésére, egy konkrét esemény részleteinek lekérdezésére és a fizetés/foglalásra is. Ezek a műveletek egy „külső” rendszerrel való összedolgozás eredményét képezik, mely rendszer az egyszerűség kedvéért a TicketApp részét képezi egy külön modulként. Ez azonban bármikor leválasztható, amennyiben a közös Commons modulból importáljuk a megfelelő adatstruktúrákat, modelleket és konfigfájlokat. Az applikáció Spring Bootban készült, az embedded TomCat szerveren fut, és a beépített H2 In-Memory Databaset használ, ami produkcióban természetesen lecserélésre kerül. Spring Data JPA került ORM Toolként használatra.

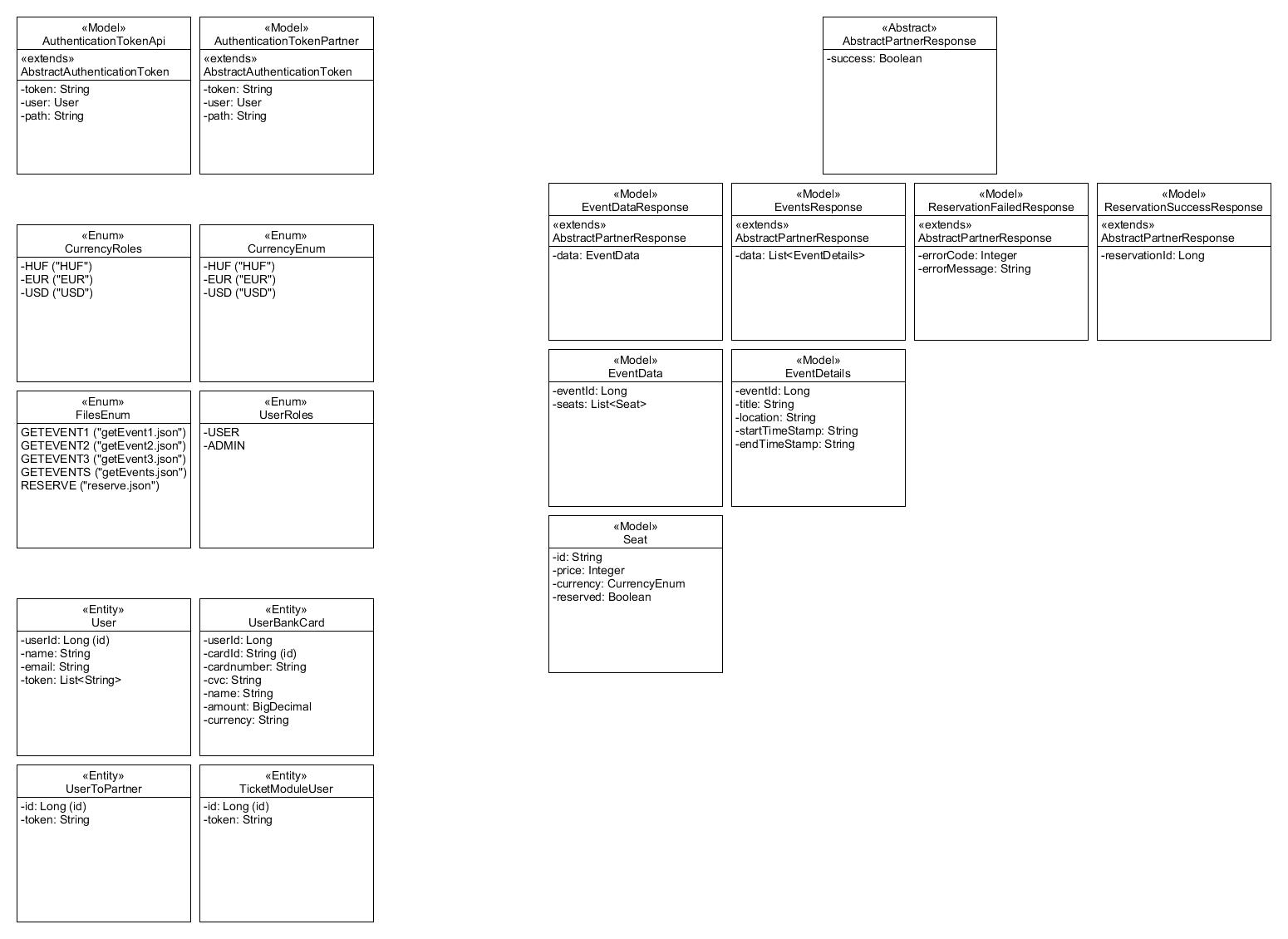
Az applikáció a következő modulokból áll:

* Appl (applikáció boot file-ja és a Spring Security konfigurációja, Custom Token Authentication)
* Commons (közös modellek, builderek, enumerációk, konstansok)
* Api (az applikáció http belépési ponjai)
* Core (belső adatbázis kezelése, abban ellenőrzések végzése, autentikációban használatos funkciók)
* Ticket (kommunikáció a Partnerrel, ellenőrzés, hogy a tranzakció lebonyolítható-e)
* Partner („külső” rendszer, amely az események adataid tartalmazza, azok kezelését végzi, valamit metódusokat biztosít a Security számára, hogy csak a Ticket modul érhesse el)

# Diagramok

Megjegyzés: a diagramok JPG formátumban a projekt főkönyvtárában találhatóak.

## Adatmodellek és entitások



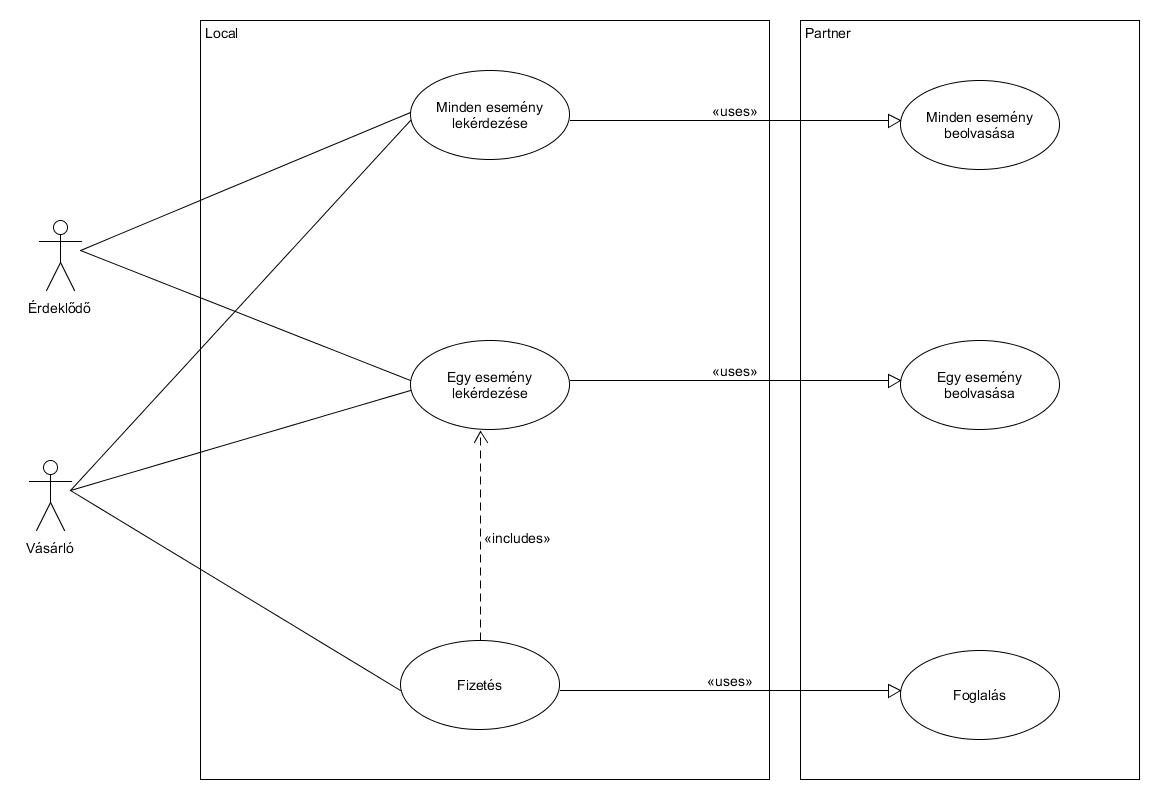
Az AuthenticationTokenApi és AuthenticationTokenPartner a Spring Security Custom Token Authentication implementációjában használatos.

A User és UserBankCard entitások a Core modul által használt adatbázistáblák entitásai, míg az UserToPartner és a TicketModuleUser entitások pedig a Ticket és Partner modul közötti autentikációhoz szükségesek, melyek egy-egy adatbázishoz tartozó entitások.

Az AbstractPartnerResponse abstract class leszármaztatott osztálya az összes response objektum, így egységesíthetővé válik az applikáció.

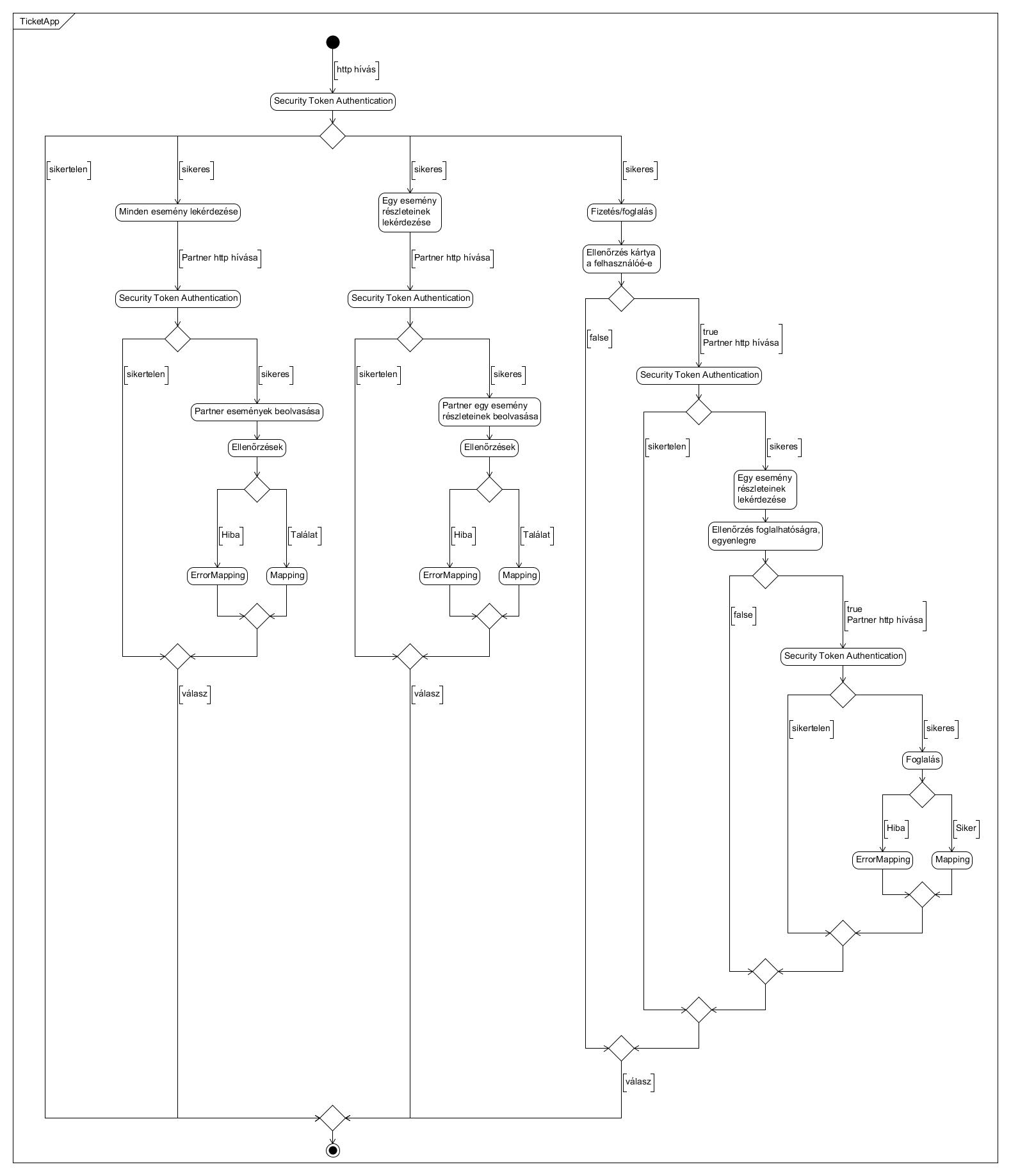
A ReservationSuccessResponse és ReservationFailedResponse objektumok építésére Builder Pattern kerül használatra.

## Use-case diagram



Alapvetően két fajta felhasználó lehetséges, az Érdeklődő, aki csupán az események, illetve konkrét esemény lekérdezésére használja az applikációt, illetve pedig a Vásárló, aki jegyet is foglal. Ez az eset viszont magában foglalja az egy konkrét esemény lekérdezését is.

## Activity diagram



Minden http hívás előtt a Spring Security Filter Chainbe beépítésre kerül egy egyedi tokenautentikációs filter, ami gondoskodik arról, hogy a megfelelő végpontokat csak autentikált felhasználók érhessék el, és információt hordoz magáról a felhasználóról, illetve biztosítja, hogy a Partner modult csak a Ticket modul érhesse el. Ha a Partnertől érkező válaszban mappolás történik, a REST végpontokban pedig hyperlink vezérlésű entitások kerülnek átadásra (Spring HATEOAS) a kliensbeli további operációk megkönnyítésére és egyszerűsítésére.

# Modulok

## Appl modul

Az Appl modul tartalmazza a Spring Boot Application starter file-t. Itt gondoskodni kell a komponensek szkenneléséről, az entitások szkenneléséről, a property source-k beállításáról, a JPA repository-k beállításáról, valamint a cache-selés és a scheduling engedélyezéséről. Továbbá a corsFilter konfigurációjáról, ami produkcióban szűkítésre kell hogy kerüljön.

Az „Adatmodellek” fejezetben említett, a Spring Securithez szükséges AuthenticationTokenApi és AuthenticationTokenPartner modellek ide kerülnek.

### AuthFilter

Egy további filtert tartalmaz mely az AbstractAuthenticationProcessingFilter absztrakt osztály leszármazott osztálya.

Implementálandó metódusok:

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | void | doFilter | ServletRequest, ServletResponse, FilterChain | IOException, ServletException |
| public | Authentication | attemptAuthentication | HttpServletRequest, HttpServletResponse | IOException, ServletException |

Gondoskodni kell a HTTPHEADER-ben „User-Token” kulcsának kinyeréséről, valamint hiányzó vagy lejárt vagy hibás token esetén az AuthenticationException megfelelő kezeléséről. Itt a kliens felé adott válasz hyperlink vezérlésű felépítése is feladat.

### AuthProvider

Implementálja az AuthenticationProvider interfacet és Beanként regisztrálni kell. Felülírja az interface függvényeit:

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | Authentication | authenticate | Authentication | AuthenticationException |
| public | boolean | supports | Class<?> | - |

Adatbázis-ellenőrzéseket kell végezni a tokenekkel. Ha a token nem szerepel a kérésben, vagy a küldött token nincs az adatbázisban, vagy szerepel az adatbázisban viszont lejárt (ott az adatbázisban, viszont van az adott tokennél korábbi sor a felhasználóhoz társítva), akkor hibát kell dobni:

Hibaüzenet: A felhasználói token nem szerepel!  
Hibakód: 10050

Hibaüzenet: A felhasználói token lejárt vagy nem értelmezhető!  
Hibakód: 10051

Hibaüzenet: A partner token nem szerepel!  
Hibakód: 90050

Hibaüzenet: A partner token lejárt vagy nem értelmezhető!  
Hibakód: 90051

Egyébként a felhasználó autentikált.

### SecurityConfig

A Spring Security konfigurációs fájlja, kiterjeszti a WebSecurityConfigurerAdapter osztályt.

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | void | configure | HttpSecurity | Exception |

Hozzáadjuk a fent definiált AuthFiltert, a CSRF-et kikapcsoljuk és a headers frameOptions attribútumát is letiltjuk. Ezen utóbbi csak akkor szükséges, ha a h2-consolet használni szeretnénk. Valamint beállítjuk, hogy az AuthFilter milyen URI mintákat ne vegyen figyelembe.

### Data.sql

Teszteléshez itt kapnak helyet az inicializálandó rekodok.

### Application.properties

Adatbázis és logolási konfiguráció.

## Commons modul

Általános, az applikáció nagy részében vagy egészében használt modellek, builderek. Ezek exportálhatóak az egyes modulokba, így a modulok teljesen leválaszthatóak, cserélhetőek. Az „Adatmodellek és entitások” fejezetben megtalálhatóak. Itt kap helyet a hibakezelés egy részéért felelős ControllerAdvice is (IOException, ParseException).

## Api modul

Az Api modul fogadja a kliens felől érkező autentikált hívásokat és továbbítja a Core modulnak (autentikáció -> Spring Security Filter -> Appl modul).

### ApiController

| **Metódus** | **Végpont** | **Paraméterek** | **Visszatérési érték** |
| --- | --- | --- | --- |
| GET | /getEvents | - | Az események listája |
| GET | /getEvent | eventId: Long | Az adott esemény részletei |
| POST | /pay | eventId: Long  seatId: Long  cardId: Long | Ha sikeres volt a foglalás:  • Success: true  • ReservationId: Long (generált érték)  Ha nem volt sikeres a foglalás:  • Success: false  • ErrorCode: Int (hibakód) |

Visszatérési értékként AbstractPartnerResponse típus kerül visszaküldésre a kliensnek.

A /pay végpont által hívott metódusnál a RequestHeaderből kinyerjük az User-Token kulcs értékét, hogy átadjuk a Core modulnak. Így a Core modul már használhatja a tokent, hogy kinyerje a társított userId-t.

A kliens munkájának könnyítése érdekében a visszatérési értékeket kiegészítjük hyperlinkekkel, melyek a releváns végpontokra mutatnak.

Továbbá definiálunk egy cache ürítésért felelős Scheduled metódust, amihez igény szerint végpontot is társíthatunk. Az alapértelmezett fixedDelay = 60000 millisec.

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | void | evictCacheOnSchedule | - | - |

### ApiService

Továbbítja a kérést a Core modulnak.

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | AbstractPartnerResponse | getEvents | - | - |
| public | AbstractPartnerResponse | getEvent | eventId: Long |  |
| public | AbstractPartnerResponse | pay | eventId: Long  seatId: Long  cardId: Long | - |
| public | void | evictCacheOnSchedule | - | - |

## Core Modul

A Core modul felelős a felhasználó- és kártyakezelésért.

### User, UserBankCard, UserRepository, UserBankCardRepository

Az „Adatmodellek és entitások” fejezetben definiált entitások a címben említett JPARepository kiterjesztések segítségével kerülnek mentésre, betöltésre, módosításra, törlésre.

UserRepository

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | Optional<User> | findByToken | token: String | - |

UserBankCardRepository

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | Optional<UserBankCard> | findByUserIdAndCardId | token: String | - |
| public | Optional<UserBankCard> | findByCardId | cardId: String | - |

Az User osztályhoz társított Users tábla és UserToken tábla megkötései:

* userId: Long – ID, nem nullázható, generált érték generátor osztállyal, 1000, 2000 stb formátumban
* name: String – max 100 karakter
* email: String – E-mail regex pattern
* token: List<String> - ElementCollection(FetchType.EAGER), CollectionTable(name = „UserToken”, JoinColumn(name =”userId”)), generált érték generátor osztállyal

Az UserBankCard osztályhoz társított azonos nevű tábla megkötései:

* userId: Long
* cardId: String – ID, generált érték generátor osztállyal, C0001, C0002 formátumban
* cardnumber: String – hossz: 16 szám
* cvc: String – hossz: 3 szám
* name: String – hossz: max 100 karakter
* amount: BigDecimal
* currency: String (a CurrencyEnum toString() metódusának értékei)

### IncomingRequestService

Fogadja az ApiService kéréseit, továbbítja a HelperService-nek.

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | AbstractPartnerResponse | getEvents | - | - |
| public | AbstractPartnerResponse | getEvent | eventId: Long |  |
| public | AbstractPartnerResponse | pay | eventId: Long  seatId: Long  cardId: Long | - |
| public | void | evictCacheOnSchedule | - | - |

### HelperService

A Core modul mediátora, az IncomingRequestService, a DatabaseHandlerService és az OutgoingService működésének összehangolása.

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | AbstractPartnerResponse | getEvents | - | - |
| public | AbstractPartnerResponse | getEvent | eventId: Long |  |
| public | AbstractPartnerResponse | pay | eventId: Long  seatId: Long  amount: BigDecimal | - |
| public | void | evictCacheOnSchedule | - | - |

* A getEvents hívás továbbítása az OutGoingService-nek
* A getEvent hívás továbbítása az OutGoingService-nek
* A pay hívás során:
  + A DatabaseHandlerService getUserFromAuthToken metódusának meghívásával az userId kinyerése
  + A DatabaseHandlerService getIfUserIdOwnsCardId metódusának meghívásával ellenőrzés, hogy a felhasználóhoz van-e az adott kártya társítva
  + A DatabaseHandlerService getAmountFromCardId metódusának meghívásával az ügyfél egyenlegének kinyerése
  + Az OutGoingService pay metódusának meghívása, ha nem történt hiba
  + Ha innen sikeres foglalás válasz érkezik, akkor az „amount” cache-ből üríti a kérdéses cardId értékét és visszaadja az őt hívó szervíznek az objektumot
  + Ha sikertelen foglalás válasz érkezik, a hiba továbbítása az őt hívó szervíznek

Hiba dobása, ha nem az ügyfélhez tartozik a kártya:

Hibaüzenet: Ez a bankkártya nem ehhez a felhasználóhoz tartozik!  
Hibakód: 10100

* Az evictCacheOnSchedule hívás továbbítása a DatabaseHandlerService-nek

### DataBaseHandlerService

Az adatbázis műveletekért felelős szervíz.

| **Privacy** | **Return** | **Name** | **Parameters** | **Cache** |
| --- | --- | --- | --- | --- |
| public | User | getUserFromAuthToken | token: String | user |
| public | Boolean | getIfUserIdOwnsCardId | userId: Long  cardId: String | userOwnsCard |
| public | BigDecimal | getAmountFromCardId | cardId: Long | amount |
| public | void | evictCacheOnAmountWithCardId | cardId: Long | CLEARS:  amount #cardId |
| public | void | evictCacheOnSchedule | - | CLEARS:  ALL |

### OutGoingService

Kérések továbbítása a Core modulból a Ticket modulba.

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | AbstractPartnerResponse | getEvents | - | - |
| public | AbstractPartnerResponse | getEvent | eventId: Long |  |
| public | AbstractPartnerResponse | pay | eventId: Long  seatId: Long  amount: BigDecimal | - |
| public | void | evictCacheOnSchedule | - | - |

### Core.properties

Adatbázis mezők validálási üzenetei.

name.required=Name is required!  
email.required=Email is required!  
email.invalid=Email format is invalid!  
cardId.invalid=Card ID format is invalid!

## Ticket Modul

A Ticket modul kommunikál a külsős Partnerrel HTTP kapcsolaton keresztül. Fizetés során a Core modultól kapja meg az információt hogy az adott tranzakció lebonyolítható-e.

### UserToPartner, UserRepositoryToPartner

Entitás a UserRepositoryToPartner JPARepository kiterjesztéshez.

UserToPartner

Az „Adatmodellek és entitások” fejezetben található a definiálás.

Speciális megkötések az adattáblához:

* id: Long – ID, nem nullázható, generált érték automata inkrementáló generálás
* token: String

UserRepositoryToPartner

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | Optional<UserToPartner> | findById | id: Long | - |

### TicketDatabaseCallerService

Adatbázis hívás a partnerhez szóló User-Token értékért.

| **Privacy** | **Return** | **Name** | **Parameters** | **Cache** |
| --- | --- | --- | --- | --- |
| public | String | getToken | - | partnerToken |
| public | void | evictCacheOnSchedule | - | CLEARS:  ALL |

### TicketBuyingService

Fogadja a Core modul hívásait, előkészíti a Partner hívásokat, valamint a Partnertől visszatérő értékeken utómunkálatokat végez.

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | AbstractPartnerResponse | getEvents | - | - |
| public | AbstractPartnerResponse | getEvent | eventId: Long |  |
| public | AbstractPartnerResponse | pay | eventId: Long  seatId: Long | - |
| private | AbstractPartnerResponse | mapPartnerErrorsToCoreErrors | AbstractPartnerResponse | - |

* A getEvents hívás továbbítása a PartnerCallerService-nek
* A getEvent hívás továbbítása az PartnerCallerService-nek
* A pay hívás során:
  + A getEvents metódusának meghívásával és egy mappeléssel az aktuális EventDetails objektumból a startTimeStamp kinyerése, ennek ellenőrzése
  + A getEvent metódusának meghívásával és egy mappeléssel az aktuális EventDataResponse objektumból a vonatkozó Seat objektum reserved értékének kinyerése, ennek ellenőrzése
  + A PartnerCallerService pay metódusának meghívása
  + Válasz mappelése

Hiba dobása a következő esetekben:

Hibaüzenet: Nem létezik ilyen esemény!  
Hibakód: 20001

Hibaüzenet: Olyan eseményre, ami már elkezdődött, nem lehet jegyet eladni!  
Hibakód: 20011

Hibaüzenet: A felhasználonak nincs elegendő pénze hogy megvásarolja a jegyet!  
Hibakód: 10101

Hibaüzenet: A külső rendszer nem elérhető!  
Hibakód: 20404

Partnertől érkező hibák mappolása a következő esetekben:

Partnerhiba: 90001  
Hibaüzenet: Nem létezik ilyen esemény!  
Hibakód: 20001

Partnerhiba: 90002  
Hibaüzenet: Nem létezik ilyen szék!  
Hibakód: 20002

Partnerhiba: 90010  
Hibaüzenet: Már lefoglalt székre nem lehet jegyet eladni!  
Hibakód: 20010

Partnertől érkező hibák továbbítása a következő esetekben:

Hibaüzenet: A partner token nem szerepel!  
Hibakód: 90050

Hibaüzenet: A partner token lejárt vagy nem értelmezhető!  
Hibakód: 90051

### PartnerCallerService

A Partner Modullal való kommunikáció HTTP-n keresztül. Erre a célra RestTemplate kerül használatra.

A HEADER-be a „User-Token” kulcs értékeként a modul adatbázisából kinyert partnertokent helyezzük az autentikáció sikeressége érdekében.

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | AbstractPartnerResponse | getEvents | - | - |
| public | AbstractPartnerResponse | getEvent | eventId: Long |  |
| public | AbstractPartnerResponse | reserve | eventId: Long  seatId: Long | - |

A RestTemplate exchange metódusával visszatérő Partner válasz Stringbe kerül mappelésre.

Ha ez null, akkor hiba dobása. Dobott hiba:

Hibaüzenet: A külső rendszer nem elérhető!  
Hibakód: 20404

Ha a válasz nem tartalmazza a ’„success”:true’ substringet, Gson használatával a Stringet ReservationFailedResponse objektummá mappeljük.

Ha pedig tartalmazza a ’„success”:true’ substringet, Gson használatával a Stringet EventsResponse/EventDataResponse/ReservationSuccessResponse objektummá mappeljük a hívástól függően (getEvents/getEvent/reserve).

Az esetleges autentikációs hibák miatt a HttpClientErrorException elkapása szükséges, amit a megfelelő hibává mappelünk.

Ha a betöltendő JSON nem áll rendelkezésre a Partner modulban, akkor a HttpServerErrorExceptiont elkapva a következő hibát dobjuk:

Hibaüzenet: Nem létezik az események tárolója!  
Hibakód: 91404

## Partner modul

Egy többnyire statikus jegykezelő rendszer, ami JSON fájlokból olvassa be az adatokat. És In-Memory H2 Adatbázist használ a TokenAutentikáció biztosításához (Security fájlok az Appl modulban). Az In-Memory DB produkcióban lecserélésre kerül.

### TicketModuleUser, TicketModuleUserRepository

TicketModuleUser

Az entitás definíciója az „Adatmodellek és entitások” fejezetben található.

Speciális megkötések az adattáblához:

* id: Long – ID, nem nullázható, generált érték automata inkrementáló generálás
* token: String

TicketModuleUserRepository

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | Optional<TicketModuleUser> | findById | id: Long | - |
| public | Optional<TicketModuleUser> | findByToken | token: String | - |

### PartnerDatabaseCallerService

Adatbázis hívás a Ticket modultól érkező User-Token értékért.

| **Privacy** | **Return** | **Name** | **Parameters** | **Cache** |
| --- | --- | --- | --- | --- |
| public | String | getToken | - | partnerUser |
| public | void | evictCacheOnSchedule | - | CLEARS:  ALL |

### PartnerController

Fogadja a Ticket modultól érkező hívásokat, továbbítja azokat a PartnerService-nek.

| **Metódus** | **Végpont** | **Paraméterek** | **Visszatérési érték** |
| --- | --- | --- | --- |
| GET | /getEvents | - | Az események listája |
| GET | /getEvent | eventId: Long | Az adott esemény részletei |
| POST | /reserve | eventId: Long  seatId: Long | Ha sikeres volt a foglalás:  • Success: true  • ReservationId: Long (generált érték)  Ha nem volt sikeres a foglalás:  • Success: false  • ErrorCode: Int (hibakód) |

A metódusok IOExceptiont és ParseExceptiont dobnak, amelyeket a Commons modulban deklarált ControllerAdvice kap el és továbbít hiba formájában a Ticket modulnak.

### PartnerService

Végzi a kérések végfeldolgozását, a megfelelő adatoknak, a megfelelő JSON fájlokból való beolvasását.

| **Privacy** | **Return** | **Name** | **Parameters** | **Throws** |
| --- | --- | --- | --- | --- |
| public | AbstractPartnerResponse | getEvents | - | IOException,  ParseException |
| public | AbstractPartnerResponse | getEvent | eventId: Long | IOException,  ParseException |
| public | AbstractPartnerResponse | makeReservation | eventId: Long  seatId: Long | IOException,  ParseException |
| public | void | evictCacheOnSchedule | - | - |

* A getEvents és a getEvent metódus beolvassa a kérdéses JSON fájlok tartalmát és azok tartalmát EventsResponse/EventDataResponse objektumokká mappeli és visszaadja, ha hiba történik a beolvasáskor akkor a hibát mappeli és továbbadja
* A makeReservation metódus:
  + Beolvassa az adott eseményt, ha ez sikertelen hibát dob
  + Ellenőrzi, hogy van-e ilyen szék, ha nincs, hibát dob
  + Ellenőrzi, hogy az adott szék foglalt-e, ha igen, hibát dob
  + Egyébként visszaad egy ReservationSuccessResponse objektumot, egy generált reservationId-vel.
* Az evictCacheOnSchedule törli a cache-selt adatokat

Dobható hibák:

Hibaüzenet: Nem létezik ilyen esemény!  
Hibakód: 90001

Hibaüzenet: Nem létezik ilyen szék!  
Hibakód: 90002

Hibaüzenet: Már lefoglalt székre nem lehet jegyet eladni!  
Hibakód: 90003

Hibaüzenet: Nem létezik az események tárolója!  
Hibakód: 91404

# Függelék

## További tervezett munkálatok

* TODO: Sikeres foglalás esetén levonja az egyenlegből a jegy árát és frissíti az adatbázist
* TODO: Sikeres foglalás esetén a megfelelő Seat objektumban a reserved booleant true-ra állítja, törlés esetén false-ra
* TODO: Külső vagy belső valutakonverter beépítése