# Kapcsolattartók dokumentáció

## Tartalomjegyzék:

# Felhasználói végpontok:

(Minden felhasználói végpont teszteléséhez a Postman alkalmazás lett használva)

## Listázás felhasználói eset:

Az oldalon megjeleníthető az összes olyan kapcsolattartót, aki aktív, őket ABC sorrendben. Először csak az első 10 személy jelenik meg(http://localhost:8080/contacts), utána pedig az oldalszám megadásával újabb 10 kapcsolattartó(http://localhost:8080/contacts?page=). Ezen az oldalon egy kapcsolattartónak csak a teljes neve, e-mail címe, telefonszáma és cégének neve van megadva.

## Részletezés felhasználói eset:

A felhasználó továbbá el tudja érni bármelyik kapcsolattartót az id-ja alapján, majd meg is tudja nézni a keresett személy összes adatát. Az utóbbi eredményt a http://localhost:8080/contact/id oldalon tudja elérni, ahol az id egy szám, ami kapcsolattartótól függő.

## Létrehozás felhasználói eset:

A felhasználó fel tud venni új kapcsolattartót, ahol a következő adatokat kell kötelezően megadnia:

Kapcsolattartó vezetékneve, keresztneve, e-mail címe, cégének nevét és státuszát. Továbbá opcionálisan meg lehet még adni a kapcsolattartó telefonszámát, kommentjét.

## Módosítás felhasználói eset:

A usernek lehetősége van egy kapcsolattartó bármelyik adatának megváltoztatására, itt először betöltődik a kiválasztott kapcsolattartó és annak minden adata, és ennek a személynek lehet majd bármelyik adatát átírni, majd a megfelelő mezők frissülni fognak az adatbázisban is. A funkciót ezen az elérési útin lehet használni, http://localhost:8080/contact/update/id, ahol az id a kiválasztott kapcsolattartó id-ja.

## Törlés felhasználói eset:

Ezzel a funkcióval a user egy általa választatott kapcsolattartónak a státuszát tudja ACTIVE-ról DELETED-re változtatni, ezzel az adatbázisból még nem törlődik a személy viszont, a fő oldalon már nem fog megjelenni. Ezt a felhasználó a http://localhost:8080/contact/deactivate/id oldalon tudja elérni, ahol az id a megadott kapcsolattartó id-ja.

# Adatbázisok:

## Company tábla:

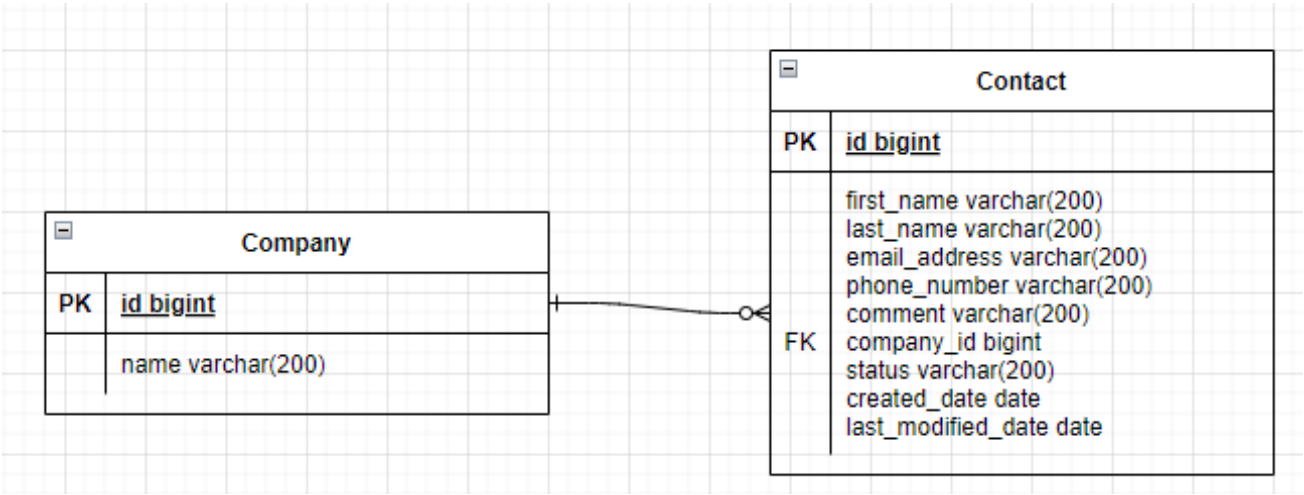
| Név | Típus | Kikötés |
|---|---|---|
| Id | Long | Nem lehet null |
| Name | String | Nem lehet null |

## Contact tábla:

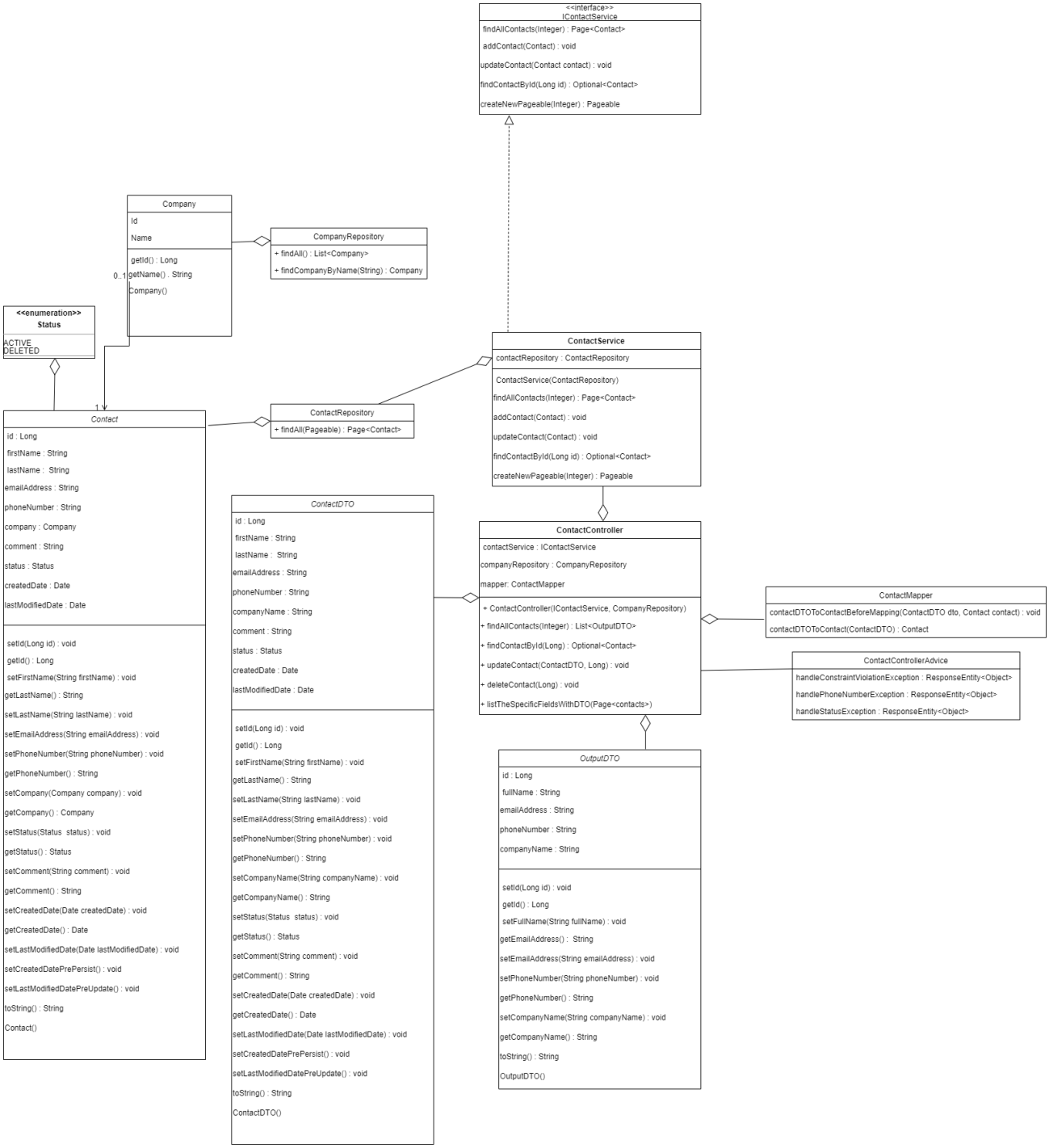
| Név | Típus | Kikötés |
|---|---|---|
| Id | Long | Nem lehet null |
| firstName | String | Nem lehet null, nem lehet üres String |
| lastName | String | Nem lehet null, nem lehet üres String |
| phoneNumber | String | Opcionális, E-164szabványnak megfelelő |
| emailAddress | String | Nem lehet üres, E-Mail cím formátumú legyen |
| company | Hozzárendelés | Nem lehet üres, Kiválasztható egy darab cég az adatbázisból |
| status | Enumeráció | Nem lehet üres, ACTIVE vagy DELETED |
| comment | String | |
| createdDate | Date | |
| lastModifiedDate | Date | |

### Diagrammok:

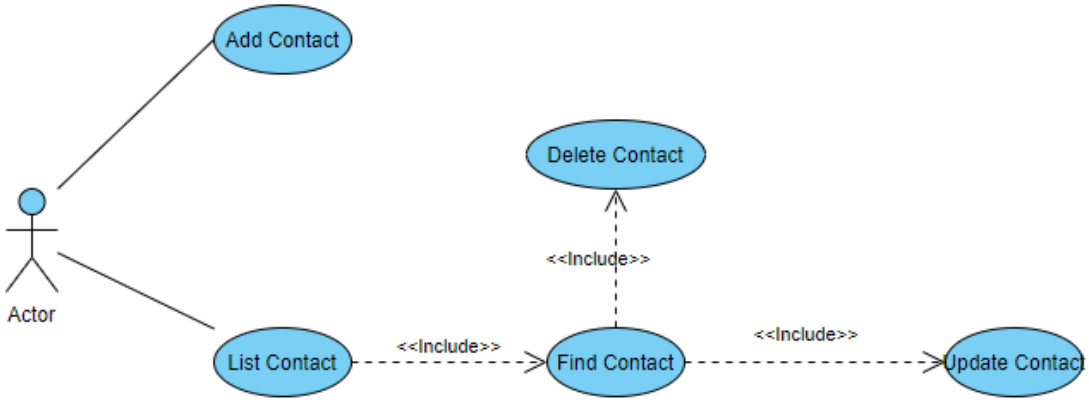
## Egyedkapcsolat diagramm:



# Osztálydiagramm:



## Use Case diagramm:




# Tesztelés:

## Service test:

1. **addValidContactTest():** A tesztelés során a ContactRepository segítségével felveszünk egy új kapcsolattartót az adatbázisba, itt lekéjük az adatbázis méretét a beillesztés előtt és után is.
2. **addInvalidContactTest():** Ebben a tesztben megpróbálunk felvinni az adatbáiba egy rosszul kitölött kapcsolattartót. Itt a várt eredmény egy ConstraintViolationException.
3. **findAllContactsTest():** A tesztben a mockolt ContactRepository lett használva a ContactService tesztelésére, azon belül is a findAll() függvény lett tesztelve.
4. **updateExistingContactTest():** Ebben a teszt esetben egy előre létrehozott kapcsolattartót vettünk fel az adatbázisba(input), majd firstname-jét és lastname-jét megváltoztattuk a ContactService updateContact függvényével, és elmentettük a változtatásokat az adatbázisba. Majd a frissített kapcsolattartót lekérjük(output) és ellenőrizük az új adatokat.
5. **updateExistingContactWithBlankDataTest()**: A tesztben létrehozunk egy új Contactot(input), majd megpróbáljuk beállítani a vezetéknevét és keresztnevét a kapcsolattartónak üres Stringre, ekkor a várt eredmény egy ConstraintViolationException.
6. **findByValidIDTest():** Ebben a tesztesetben először létre lett hozva egy kapcsolattartó(input), majd el lett mentve az adatbázisban. Az ellenőrzéshez a findContactByID metódus lett használva, aminek paraméterül az input Id-ját adtuk meg, ez visszatér egy kapcsolattartóval(output). A teszteset végén az input és az output Contact kötelező mezői összehasonlításra kerülnek.
7. **findByInvalidIdTest():** A teszt során egy olyan Id-jú Contactra próbálunk rákeresni, aki nem létezik, ekkor az elvárt eredmény egy ConstraintViolationException.

## Controller test:

I. **blankFirstNameTest():** A Contact validálás lett tesztelve, azon belül az, hogy egy Contact felvételénél vagy módosításánál nem lehet üres adatot átadni. A teszt során létrehoztunk egy jól kitöltött kapcsolattartót, de a perzisztálás előtt átírtuk a vezetéknevét egy üres Stringre, ezután az elvárt eredmény egy ConstraintViolationException, “Vezetéknév nem lehet üres” üzenettel és 409-es kóddal.

II. **nullFirstNameTest():** A Contact validálás lett tesztelve, azon belül az, hogy egy Contact felvételénél vagy módosításánál muszáj bizonyos mezőket kitölteni. A teszt során létrehoztunk egy jól kitöltött kapcsolattartót, de a perzisztálás előtt a vezetéknevét nullra állítottuk, ezután az elvárt eredmény egy ConstraintViolationException, “Vezetéknév kitöltése kötelező” üzenettel és 409-es kóddal.

III. **invalidEmailAddressTest()**: Ezzel a tesztel lesz bemutatva, hogy mi történik akkor, ha rossz a kitöltött e-mail cím formátuma. A teszt elején létrehozunk egy jól kitöltött kapcsolattartót és a perzisztálás előtt átírjuk az e-mail cím formátumát olyanra, amit biztosan nem fog elfogadni a program, ekkor a várt eredmény egy ConstraintViolationException, “Helyes formátumú e-mail címnek kell lennie” üzenettel és 409-es hibakóddal.

IV. **invalidPhoneNumberTest():** Ezzel a tesztel lesz bemutatva, hogy mi történik akkor, ha rossz a kitöltött telefonszám formátuma. A teszt elején létrehozunk egy jól kitöltött kapcsolattartót és a perzisztálás előtt átírjuk az telefonszámot olyan formátumra, amit biztosan nem fog elfogadni a program, ekkor a várt eredmény egy NumberFormatException, “A telefonszámnak helyes formátumúnak kell lennie” üzenettel és 409-es hibakóddal.

V. **invalidCompanyTest():** Ezzel a tesztel lesz bemutatva, hogy mi történik akkor, ha a kitöltött cégnév még nem létezik. A teszt elején létrehozunk egy jól kitöltött kapcsolattartót és a perzisztálás előtt átírjuk a cégnevét olyanra, amilyen még nem létezik, ekkor a várt eredmény egy ConstraintViolationException, “Meg kell adni a cég nevét” üzenettel és 409-es hibakóddal.

VI. **invalidStatus():** Ezzel a tesztel lesz bemutatva, hogy mi történik akkor, ha a kitöltött státusz nem létezik vagy nincs kitöltve. A teszt elején létrehozunk egy jól kitöltött kapcsolattartót és a perzisztálás előtt átírjuk a státuszát olyanra, amilyen még nem létezik, ekkor a várt eredmény egy IllegalArgumentException, “Nem létező státusz lett beállítva, válasszon egyet a következőek közül: ACTIVE, DELETED” üzenettel és 409-es hibakóddal.

VII. **deleteContactTest():** A teszt során létre lett hozva egy helyesen kitöltött kapcsolattartó majd perzisztálás után megváltoztatjuk egy PutMappinggel a státuszát DELETED-re, majd egy GetMappinggel ellenőrizzük, hogy megváltozott-e a státusz a megadott Contacton.

VIII. **updateContactTest():** A teszt során létre lett hozva egy helyesen kitöltött kapcsolattartó majd perzisztálás után megváltoztatjuk egy PutMappinggel a vezetéknevét, majd egy GetMappinggel ellenőrizzük, hogy megváltozott-e a vezetéknév a megadott Contacton.

IX. **getAllContactTest():** A teszt során létre lett hozva két helyesen kitöltött kapcsolattartó majd perzisztálás után lekérjük egy GetMappinggel az összes Contactot és ellenőrizzük, hogy jó helyre kerültek be a kapcsolattartók.

X. **findContactByRealIdTest():** A teszt során létre lett hozva egy helyesen kitöltött kapcsolattartó majd perzisztálás után az Id-ja alapján megpróbáljuk lekérni ezt a személyt az adatbázisból.

**XI.** **findContactByFakeIdTest():** A teszt során létre lett hozva egy helyesen kitöltött kapcsolattartó majd perzisztálás után egy nem létező Id-val megpróbálunk keresni egy Contactot, ami null-t fog visszaadni.

XII. **addValidContactTest():** A teszt során létre lett hozva egy helyesen kitöltött kapcsolattartó majd elmentjük az adatbázisba, ekkor egy Ok visszatérésre számítunk.