|  |  |
| --- | --- |
| KÃ©ptalÃ¡lat a kÃ¶vetkezÅre: âyacht portraitâ  Yacht klub | **Yacht Klub DOKUMENTÁCIÓ**  YachtKlub adminisztrációs alkalmazás dokumentáció  **Készítette:**  Martina Dániel, Pörzsölt Krisztián, Kamenár Dávid, Köblös Krisztián |

Tartalom

[Feladat 2](#_Toc512893215)

[Rendszerkövetelmények 2](#_Toc512893216)

[Alapötlet 2](#_Toc512893217)

[Adatbázis 3](#_Toc512893218)

[Tagok tábla 3](#_Toc512893219)

[Szallitoeszkozok tábla 3](#_Toc512893220)

[Hajok tábla 3](#_Toc512893221)

[Ajanlat\_keres tábla 3](#_Toc512893222)

[Hajo\_kolcsonzes tábla 3](#_Toc512893223)

[Relációséma 4](#_Toc512893224)

[Fontosabb kódrészek 4](#_Toc512893225)

[Entity (nincs egyszerűsítve) 4](#_Toc512893226)

[Data Access Object (nincs egyszerűsítve) 5](#_Toc512893227)

[Validator (egyszerűsített) 6](#_Toc512893228)

[EmailFormatValidator 6](#_Toc512893229)

[NumberFormatValidator 6](#_Toc512893230)

[Használat közben 7](#_Toc512893231)

[Service (saját) 7](#_Toc512893232)

[Megvalósított funkciók 7](#_Toc512893233)

[Működés közben 7](#_Toc512893234)

[Belépés 7](#_Toc512893235)

[Főmenü 7](#_Toc512893236)

[Admin 7](#_Toc512893237)

[Felhasználó 7](#_Toc512893238)

[Hajó foglalás 8](#_Toc512893239)

[Új hajó felvétele 8](#_Toc512893240)

[Új szállítóeszköz felvétele 8](#_Toc512893241)

[Hajóim és eszközeim 8](#_Toc512893242)

[Foglalások megtekintése 8](#_Toc512893243)

# Feladat

Készíteni kellett egy grafikus felületű asztali alkalmazást, amely egy Yacht Klub adminisztrációs rendszerét valósítja meg.

Adatbázisban kellett tárolni a klubtagok és hajók adatait és kölcsönzéseket. Főbb megvalósítandó funkciók:

* új klubtag rögzítése
* új hajó rögzítése
* új szállítóeszköz rögzítése
* kölcsönzések tárolása
* bérlő adatainak megtekintése
* bérlendő hajó visszaigazolása
* megtett hajóutak rögzítések, hajók tartózkodásának rögzítése
* adminisztrátor és klubtag jogosultság létrehozása
* kimutatások elemzése

Megkötések:

* WPF technológia
* menüvezérelt felépítés
* UserControl használata
* Diagram esetén oszlopdiagram megvalósítása
* MS SQL adatbázis, 3NF
* EntityFramework használata (CodeFirst)
* LINQ vagy tárolt eljárások használata

# Rendszerkövetelmények

* Windows 7, 8, 10 64-bit operációs rendszer
* Legalább 50 MB memória legalább

# Alapötlet

A közös munka érdekében GitHub tárhelyet hoztunk létre a projektnek, melyet először egy SmartGit klienssel kötöttünk össze, később áttértünk a Git Desktop kliensére.

A kód átláthatósága érdekében pedig Design mintákat használtunk fel, melyek egymással kapcsolatban álltak. A következő diagram az osztályok kapcsolatait írja le:

A legalacsonyabb szinten az Entity osztályok állnak. Ezek valósítják meg az adatbázist. Az adatbázis leírása CodeFirst technológiával történt.

A következő szinten DAO osztályok kommunikálnak az entitás osztályokkal. Saját metódusaikkal kérdezik le az entitás osztályokból az adatokat, LINQ lekérdezések segítségével.

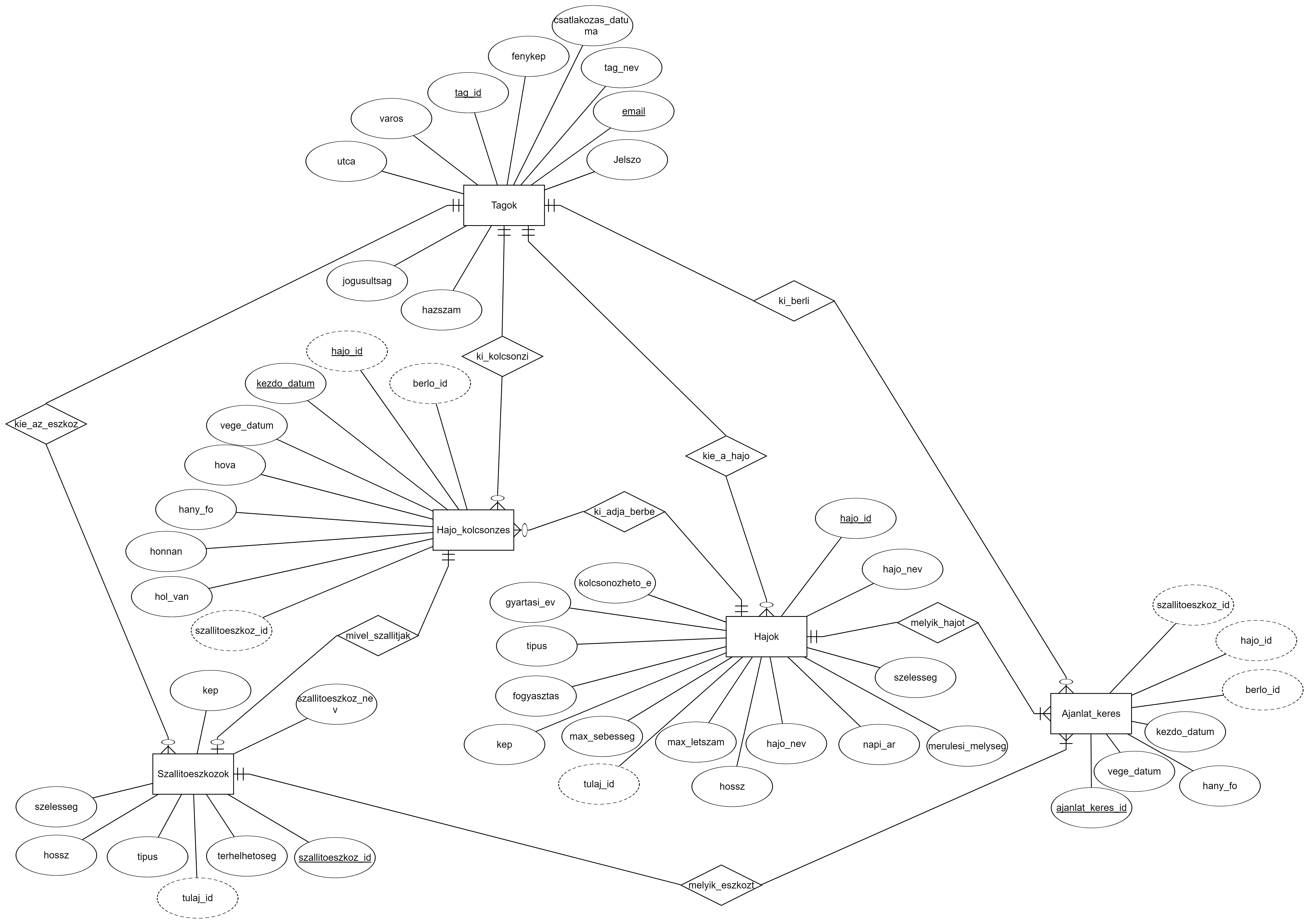
A Service osztályok valósítják meg a kód backend részét, a mögöttes logikát. Ezen osztályok használják a DAO objektumokat. Adatfeltöltés esetén azonban előfordulhat, hogy először egy entitást példányosítunk, melyet feltöltünk és átadunk a DAO objektumnak. Az Entity Framework kontextusa ez utóbbi műveletet azonban megkönnyítette azzal, hogy az entitást a Context objektumnak adtuk át, amely az adatbázissal közvetlen kapcsolatban áll. A Context objektum SaveChanges() metódusa pedig végrehajtotta a módosításokat.

A legmagasabb szinten a XAML és a felületet megvalósító .cs fájlok állnak. A XAML fájlok csak a felületleírást tartalmazzák. A .cs osztályok pedig csak a felület leírását és annak dinamikus működését hajtják végre. Itt használjuk a Validator osztályokat is, melyek az adatok helyességét ellenőrzik le, a megadott beviteli mezőknél. Ez azért jó, mert hiba esetén, például ha egy szám mezőben betű szerepel, akkor a Service objektumok nem jönnek létre, így az adatbázis lekérdezések nem valósulnak meg – mivel egyébként is hibára futnának.

Ellenkező esetben továbbhaladunk a láncon és a Service osztályok végrehajtják az adatbázis ellenőrzését. Ha minden feltétel megfelel, akkor mind a Validator, mint a Service objektumok visszatérnek egy OK státusszal.

# Adatbázis

Az adatbázist CodeFirst technológiával alakítottuk ki. A programkód létrehozza az alap lokális adatbázist, ha az még nem létezik, ellenkező esetben csatlakozik a meglévő adatbázishoz.



## Tagok tábla

Tárolja a tagok adatait, szállítóeszközeit, hajóit és kölcsönzéseit.

## Szallitoeszkozok tábla

A szállítóeszközök adatait tárolja.

## Hajok tábla

A hajók adatait tárolja, illetve, hogy mikor kölcsönözték őket, ezáltal az is megállapítható, hogy jelenleg hol van az adott Yacht.

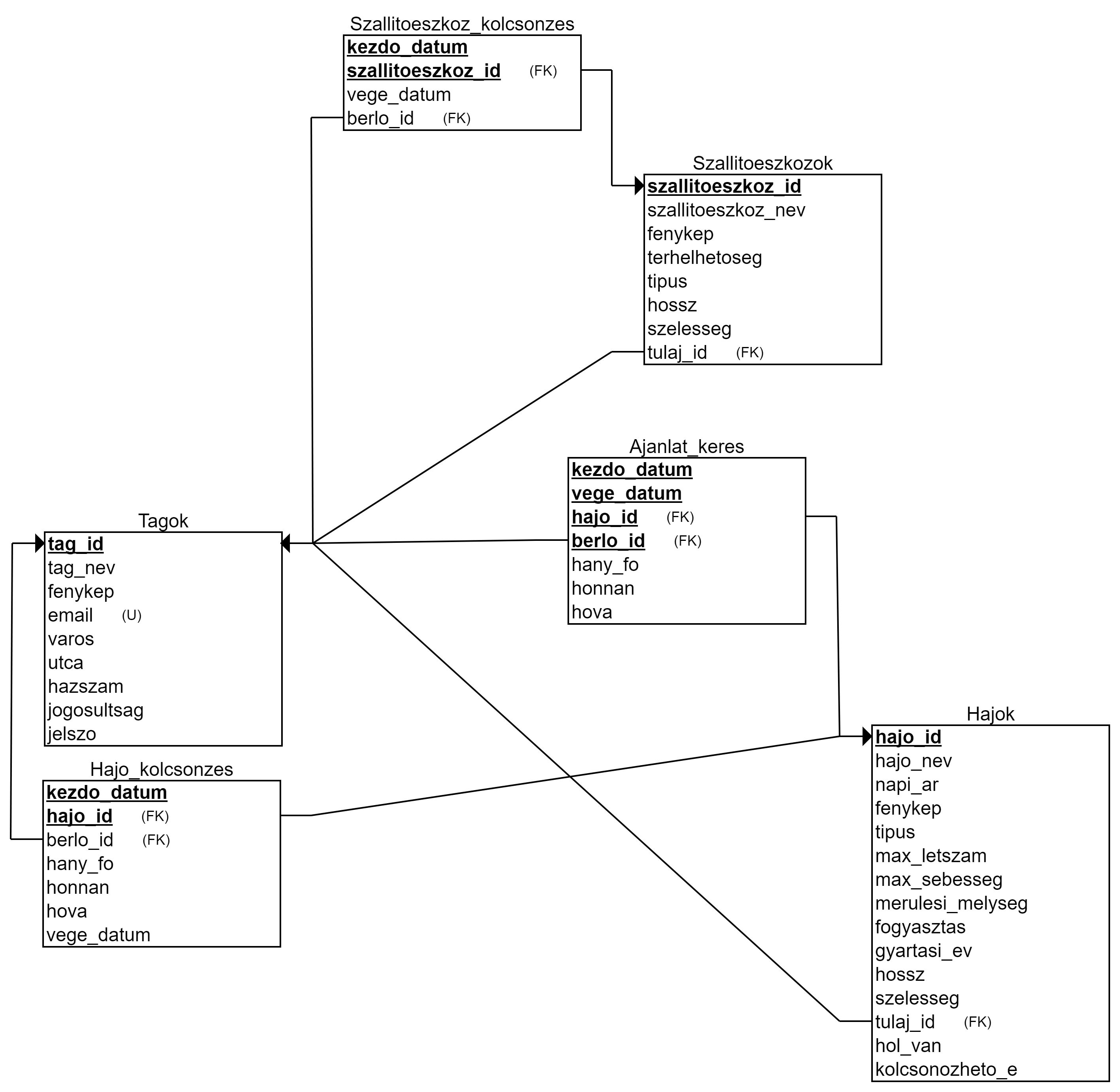
## Ajanlat\_keres tábla

Minden ajánlatkérést tárol, melyet rögzítettek a felhasználók.

## Hajo\_kolcsonzes tábla

Tárolja az összes hajó és szállítóeszköz kölcsönzést.

## Relációséma



# Fontosabb kódrészek

A kód minden része illeszkedik a fent említett mintához. Egyes részek azonban nem a komplett Design mintából lettek kialakítva. Mivel a projekt kicsi, ezért egyszerűsített osztálykapcsolatokat használ a program. Néhányat közülük alább láthatunk.

## Entity (nincs egyszerűsítve)

Minden osztály egy-egy táblának feleltethető meg az adatbázisban. A legegyszerűbb, ajánlat kérés tábla a következőképpen néz ki a C# kódban:

namespace YachtKlub.entity

{

[Table("RentRequests")]

class RentRequestsEntity

{

[Column("startingDate", TypeName = "Date", Order = 0)]

[Key]

[Required]

public DateTime StartingDate { get; set; }

[Column("endDate", TypeName = "Date", Order = 1)]

[Key]

[Required]

public DateTime EndDate { get; set; }

public BoatsEntity BoatToBorrow { get; set; }

public TransportDevicesEntity DeviceToBorrow { get; set; }

public MembersEntity WhoBorrows { get; set; }

[Column("howManyPersonWillTravel", Order = 4)]

[DataType("decimal(3, 0)")]

[Required]

public int HowManyPersonWillTravel { get; set; }

[Column("fromWhere", Order = 5)]

[StringLength(100)]

[Required]

public string FromWhere { get; set; }

[Column("toWhere", Order = 6)]

[StringLength(100)]

[Required]

public string ToWhere { get; set; }

[Column("status", Order = 7)]

[DataType("decimal(1, 0)")]

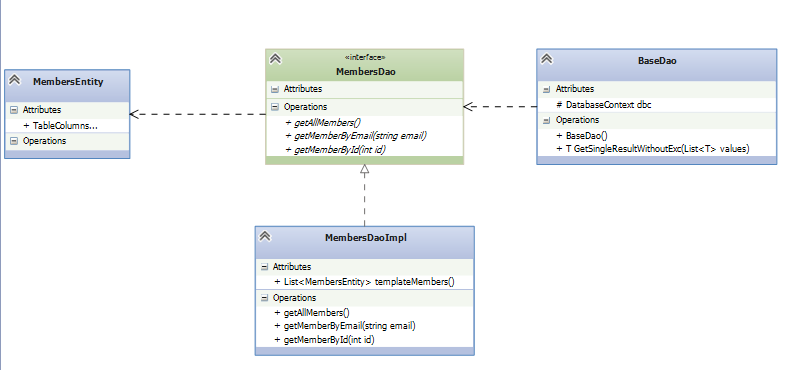
[Required]

public int Status { get; set; }

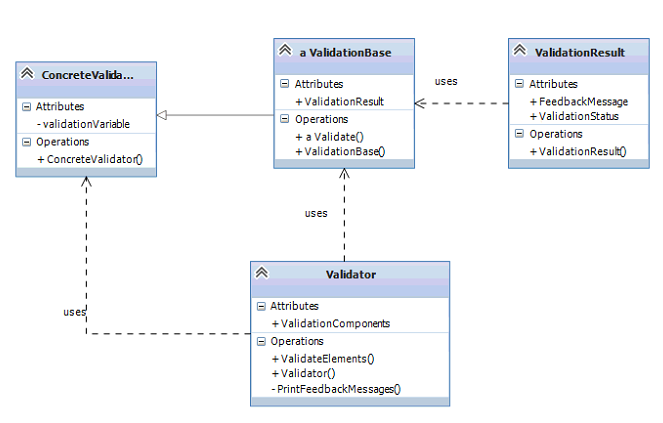
}

}

## Data Access Object (nincs egyszerűsítve)



## Validator (egyszerűsített)



## EmailFormatValidator

class EmailFormatValidator : ValidationBase

{

string email;

public EmailFormatValidator(string email)

{

this.email = email;

}

public override void Validate()

{

Regex regex = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$");

Match match = regex.Match(email);

if (!match.Success)

{

ValidationResult.ValidationStatus = Status.Error;

ValidationResult.FeedbackMessage = "Helytelen e-mail cím formátum!";

}

else

{

ValidationResult.ValidationStatus = Status.OK;

}

}

}

## NumberFormatValidator

class NumberFormatValidator : ValidationBase

{

string number;

public NumberFormatValidator(string number)

{

this.number = number;

}

public override void Validate()

{

int n;

bool isNumber = int.TryParse(number, out n);

if (!isNumber)

{

ValidationResult.ValidationStatus = Status.Error;

ValidationResult.FeedbackMessage = "Nem szám formátum egy számot elfogadó mezőben!";

}

else

{

ValidationResult.ValidationStatus = Status.OK;

}

}

}

## Használat közben

Validator loginValidator = new Validator();

loginValidator.ValidationComponents.Add(new EmptyFieldValidator(email, "e-mail cím"));

loginValidator.ValidationComponents.Add(new EmailFormatValidator(email));

loginValidator.ValidationComponents.Add(new EmptyFieldValidator(password, jelszó"));

loginValidator.ValidateElements();

## Service (saját)

Hasonló a Validator mintához.

# Megvalósított funkciók

* belépés
* regisztráció (csak admin tud regisztrálni)
* hajó foglalás (szállítóeszközzel
* saját hajóim és eszközeim megtekintése
* új hajó felvétele (csak adminnál)
* új szállítóeszköz felvétele (csak adminnál)
* profiladatok megváltoztatása
* kimutatás készítése
* foglalások megtekintése

# Működés közben

## Belépés

Ez az ablak jelenik meg először, innen lehet belépni e-mail cím és jelszó megadásával. A program automatikusan arra a felületre dob tovább, amilyen jogosultságunk van.

## 

## Főmenü

Itt lehet a különböző ablakokba továbblépni és a Yachtokat és foglalásokat kezelni.

### Admin

Az admin itt tud új felhasználót regisztrálni, valamint megváltoztathatja saját adatait és jelszavát.

### 

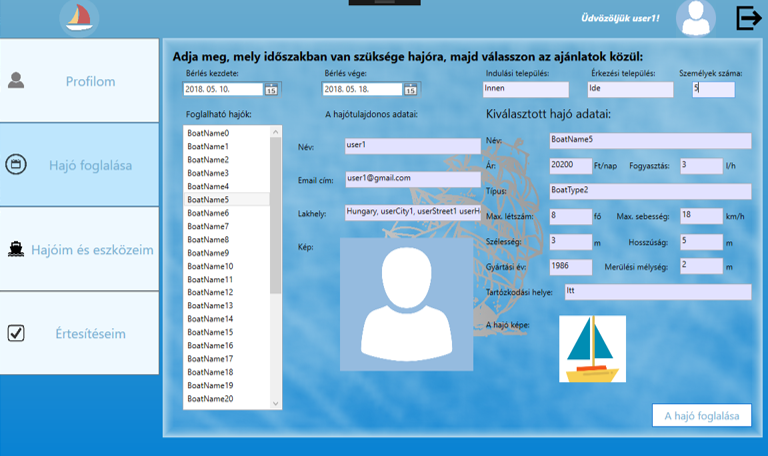
### Felhasználó

A felhasználó megváltoztathatja saját adatait és jelszavát.

## 

## Hajó foglalás

Új hajót tudunk foglalni a kiválasztott dátum szerint.



## Új hajó felvétele

Admin menüből elérhető. Új hajót vehet fel az admin.

## 

## Új szállítóeszköz felvétele

Admin menüből elérhető. Új szállítóeszközt vehet fel az admin.

## 

## Hajóim és eszközeim

Itt megtekintheti a felhasználó vagy az admin a saját hajóinak és eszközeinek adatait.

## 

## Foglalások megtekintése

Saját és mások foglalásait ­- akik tőlünk foglaltak - tekinthetjük meg.

