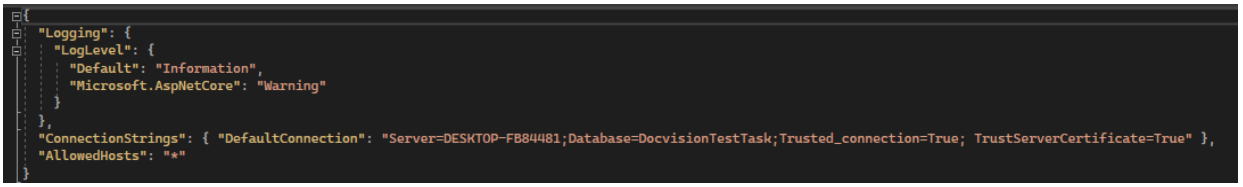
Руководство для технического специалиста. База данных.

Подключение БД:

Описание подключения к БД описано в файле appsettings.json, в корневом каталоге приложения.



```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "ConnectionStrings": { "DefaultConnection": "Server=DESKTOP-FB84481;Database=DocvisionTestTask;Trusted_connection=True; TrustServerCertificate=True" },
  "AllowedHosts": "*"
}
```

Структура БД MySQL:

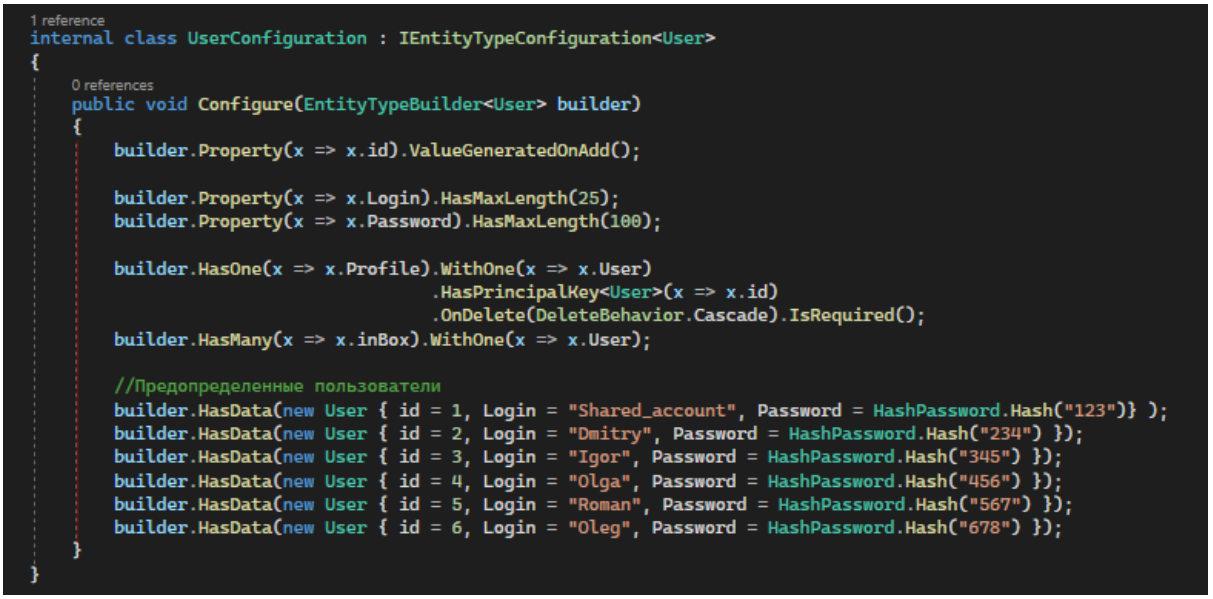
База данных API сервера представлена в виде трех таблиц:
inBox { хранение входящих писем }
Profile { хранением профилей сотрудников}
User { хранение аккаунтов сотрудников }

Таблицы имеют следующие поля и параметры:

User.

1. id - int, primary key
2. Login - nvarchar(25), логин пользователя
3. Password - nvarchar(100), пароль пользователя

Конфигурация таблицы выглядит следующим образом:



```
1 reference
internal class UserConfiguration : IEntityTypeConfiguration<User>
{
    0 references
    public void Configure(EntityTypeBuilder<User> builder)
    {
        builder.Property(x => x.id).ValueGeneratedOnAdd();

        builder.Property(x => x.Login).HasMaxLength(25);
        builder.Property(x => x.Password).HasMaxLength(100);

        builder.HasOne(x => x.Profile).WithOne(x => x.User)
                                      .HasPrincipalKey<User>(x => x.id)
                                      .OnDelete(DeleteBehavior.Cascade).IsRequired();
        builder.HasMany(x => x.inBox).WithOne(x => x.User);

        //Предопределенные пользователи
        builder.HasData(new User { id = 1, Login = "Shared_account", Password = HashPassword.Hash("123")} );
        builder.HasData(new User { id = 2, Login = "Dmitry", Password = HashPassword.Hash("234") });
        builder.HasData(new User { id = 3, Login = "Igor", Password = HashPassword.Hash("345") });
        builder.HasData(new User { id = 4, Login = "Olga", Password = HashPassword.Hash("456") });
        builder.HasData(new User { id = 5, Login = "Roman", Password = HashPassword.Hash("567") });
        builder.HasData(new User { id = 6, Login = "Oleg", Password = HashPassword.Hash("678") });
    }
}
```

Profile:

1. id - int, primary key
2. firstName - nvarchar(50), имя сотрудника
3. lastName - nvarchar(50), фамилия сотрудника
4. sureName - nvarchar(50), отчество сотрудника
5. Email - nvarchar(50), почта сотрудника
6. userId - int, связь для таблицы User

Конфигурация таблицы выглядит следующим образом:

```
1 reference
internal class ProfileConfiguration : IEntityTypeConfiguration<Profile>
{
    0 references
    public void Configure(EntityTypeBuilder<Profile> builder)
    {
        builder.Property(x => x.Id).ValueGeneratedOnAdd();
        builder.Property(x => x.firstName).HasMaxLength(50).IsRequired();
        builder.Property(x => x.lastName).HasMaxLength(50).IsRequired();
        builder.Property(x => x.sureName).HasMaxLength(50).IsRequired();
        builder.Property(x => x.Email).HasMaxLength(50).IsRequired();

        //Предопределенные профили
        builder.HasData(new Profile { Id = 1, userId = 1, firstName = "Общая почта",
            lastName = " ", Email = "dg_all@docvision.ru"});
        builder.HasData(new Profile { Id = 2, userId = 2, firstName = "Дмитрий",
            lastName = "Аносов", sureName = "Алексеевич", Email = "AnosovDA@docvision.ru" });
        builder.HasData(new Profile { Id = 3, userId = 3, firstName = "Игорь",
            lastName = "Вакин", sureName = "Дмитриевич", Email = "VakinID@docvision.ru" });
        builder.HasData(new Profile { Id = 4, userId = 4, firstName = "Ольга",
            lastName = "Микушина", sureName = "Андреевна", Email = "MikushinaOA@docvision.ru" });
        builder.HasData(new Profile { Id = 5, userId = 5, firstName = "Роман",
            lastName = "Басин", sureName = "Арсенович", Email = "BasinRA@docvision.ru" });
        builder.HasData(new Profile { Id = 6, userId = 6, firstName = "Олег",
            lastName = "Старов", sureName = "Сергеевич", Email = "StarovOS@docvision.ru" });
    }
}
```

inBox.

1. id - int, primary key
2. emailSubject - nvarchar(300), тема письма
3. emailDate - datetime2(7), дата отправки письма
4. emailFrom - nvarchar(50), отправитель письма
5. emailTo - nvarchar(50), получатель письма
6. emailBody - nvarchar(5000), тело письма
7. userId - int, связь для таблицы User

Конфигурация таблицы выглядит следующим образом:

```
1 reference
internal class inBoxConfiguration : IEntityTypeConfiguration<inBox>
{
    0 references
    public void Configure(EntityTypeBuilder<inBox> builder)
    {
        builder.Property(x => x.id).ValueGeneratedOnAdd();
        builder.Property(x => x.emailSubject).HasMaxLength(300).IsRequired();
        builder.Property(x => x.emailDate).IsRequired();
        builder.Property(x => x.emailFrom).HasMaxLength(50).IsRequired();
        builder.Property(x => x.emailTo).HasMaxLength(50).IsRequired();
        builder.Property(x => x.emailBody).HasMaxLength(5000).IsRequired();
    }
}
```

База данных имеет следующую структуру связей:

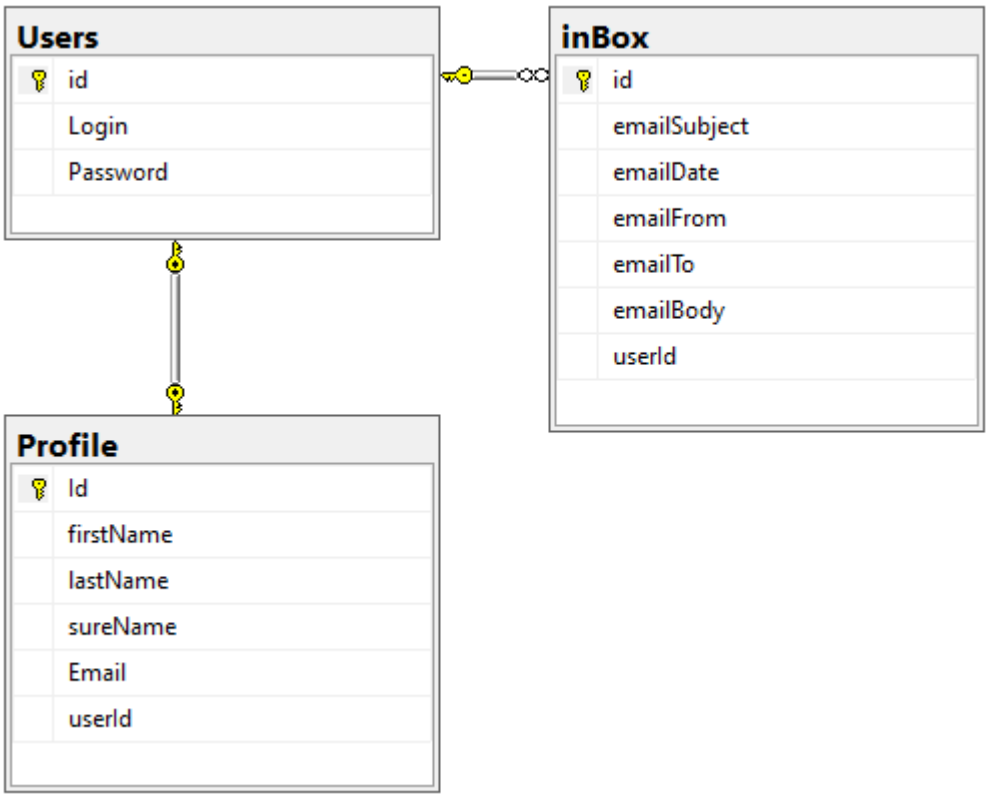


Таблица **Profile** имеет foreign key userId, для связи между таблицей **User** и **Profile** (one-to-one)
Подобная архитектура позволяет нам вынести вторичные данные такие как имя, фамилия, и т.д. в отдельную таблицу и задать жесткую привязку Один пользователь => один профиль. Т.о. мы можем использовать описание полей Profile и идентифицировать связующую запись из таблицы User.

Таблица **inBox** имеет foreign key userId, для связи между таблицей **User** и **inBox** (one-to-many). Подобная архитектура позволяет нам использовать inBox таблицу для хранения всех входящих сообщений, исключает появление “зомби” записей т.к. для каждого письма должен быть связанный с ним пользователь, а также позволяет получить все записи входящих писем, зависимые от соответствующего User.

Все поля таблиц не могут иметь значение null. Дабы исключить каскадные ошибки при дальнейшем развитие и расширения API.
Таким образом для описания моделей сущностей используется механизм стандартных значений. Описания сущностей приведены ниже.

```
20 references
public class inBox
{
    1 reference
    public int id { get; set; }
    3 references
    public string emailSubject { get; set; } = "Тема письма отсутствует";
    3 references
    public DateTime emailDate { get; set; } = DateTime.MinValue;
    3 references
    public string emailFrom { get; set; } = "Отправитель не указан";
    6 references
    public string emailTo { get; set; } = "Адресат не указан";
    3 references
    public string emailBody { get; set; } = "Отсутствует содержание письма";

    //ForeignKey
    3 references
    public int userId { get; set; }
    1 reference
    public User User { get; set; }
}
```

```
public class Profile
{
    7 references
    public int Id { get; set; }

    10 references
    public string firstName { get; set; } = "Имя не заполнено";
    10 references
    public string lastName { get; set; } = "Фамилия не заполнено";
    6 references
    public string sureName { get; set; } = "Отчество не заполнено";
    7 references
    public string Email { get; set; } = "Email не заполнен";

    //Foreign key
    7 references
    public int userId { get; set; }
    2 references
    public User User { get; set; }
}
```

```
// Описание сущности для таблицы БД User
45 references
public class User
{
    9 references
    public int id { get; set; }

    7 references
    public string Login { get; set; }
    7 references
    public string Password { get; set; }

    // Связи
    //
    // Связь для таблицы Profile
    6 references
    public Profile Profile { get; set; }

    // Связь для таблицы Inbox
    1 reference
    public ICollection<inBox> inBox { get; set; }
}
```

Описание DbContext

```
11 references
public class ApplicationDbContext: DbContext
{
    1 reference
    public DbSet<inBox> inBox { get; set; }
    3 references
    public DbSet<User> Users { get; set; }
    1 reference
    public DbSet<Profile> Profile { get; set; }
    0 references
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
    {
    }

    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder) //Вынесем конфигурацию таблиц в отдельный класс
    {
        modelBuilder.ApplyConfiguration(new inBoxConfiguration());
        modelBuilder.ApplyConfiguration(new UserConfiguration());
        modelBuilder.ApplyConfiguration(new ProfileConfiguration());
    }
}
```