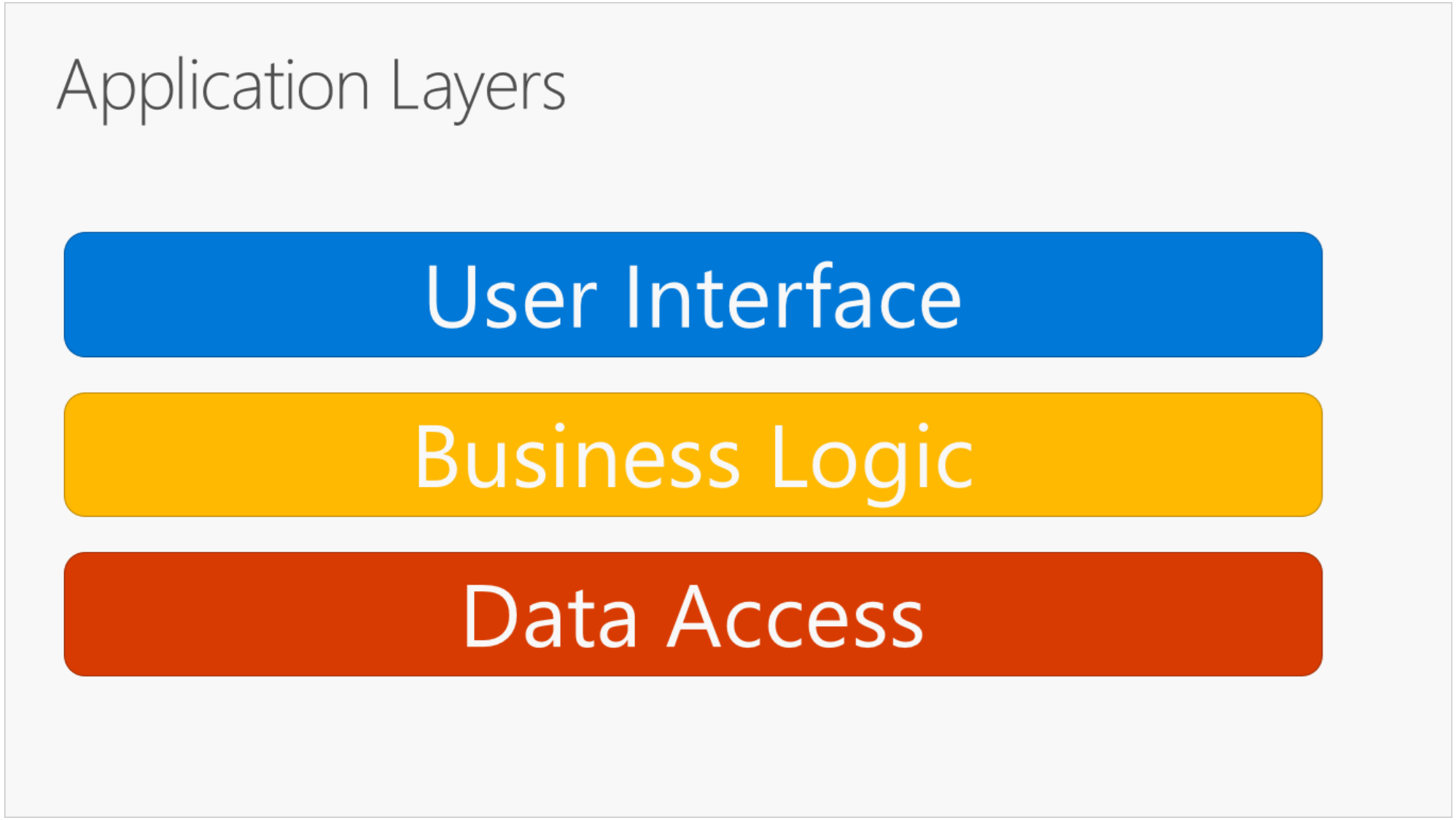
Многослойная архитектура

Многослойная архитектура

Архитектура, в которой разделяются функции **представления**, **обработки** и **хранения** данных.

Наиболее распространённой разновидностью многослойной архитектуры является трехслойная архитектура.

Зачем?

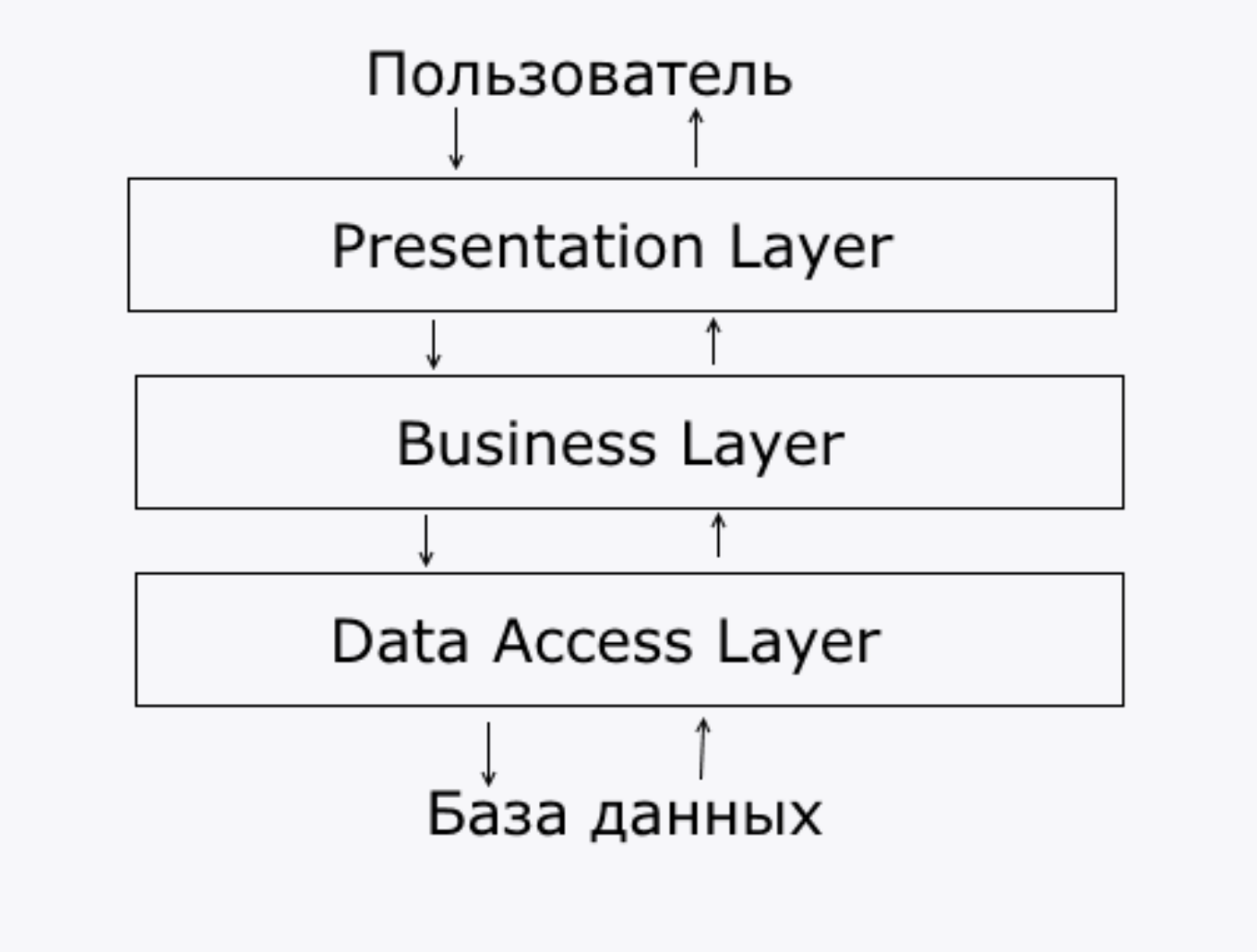

Application Layers
User Interface
Business Logic
Data Access

Многослойная vs многоуровневая

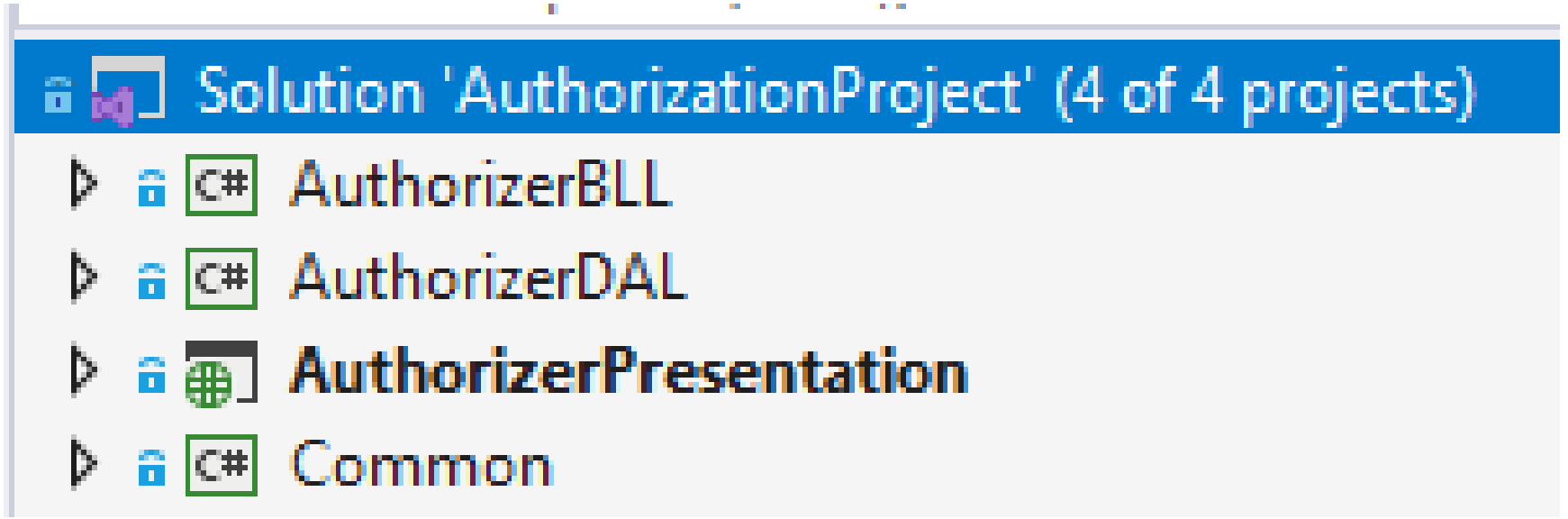
Слой — это механизм логического структурирования компонентов.

Уровень — это механизм физического структурирования инфраструктуры системы.

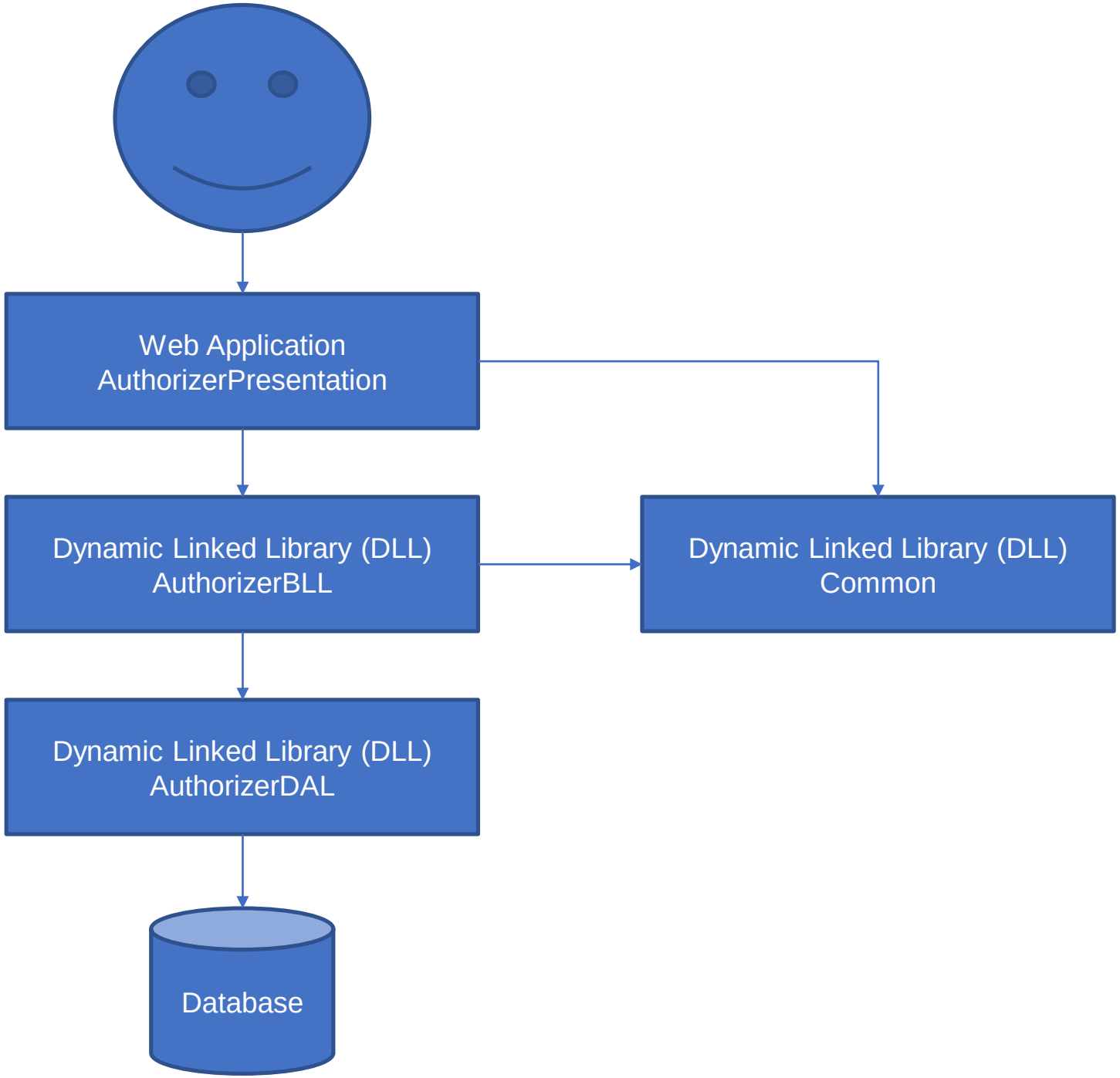
Классическая трехзвенка



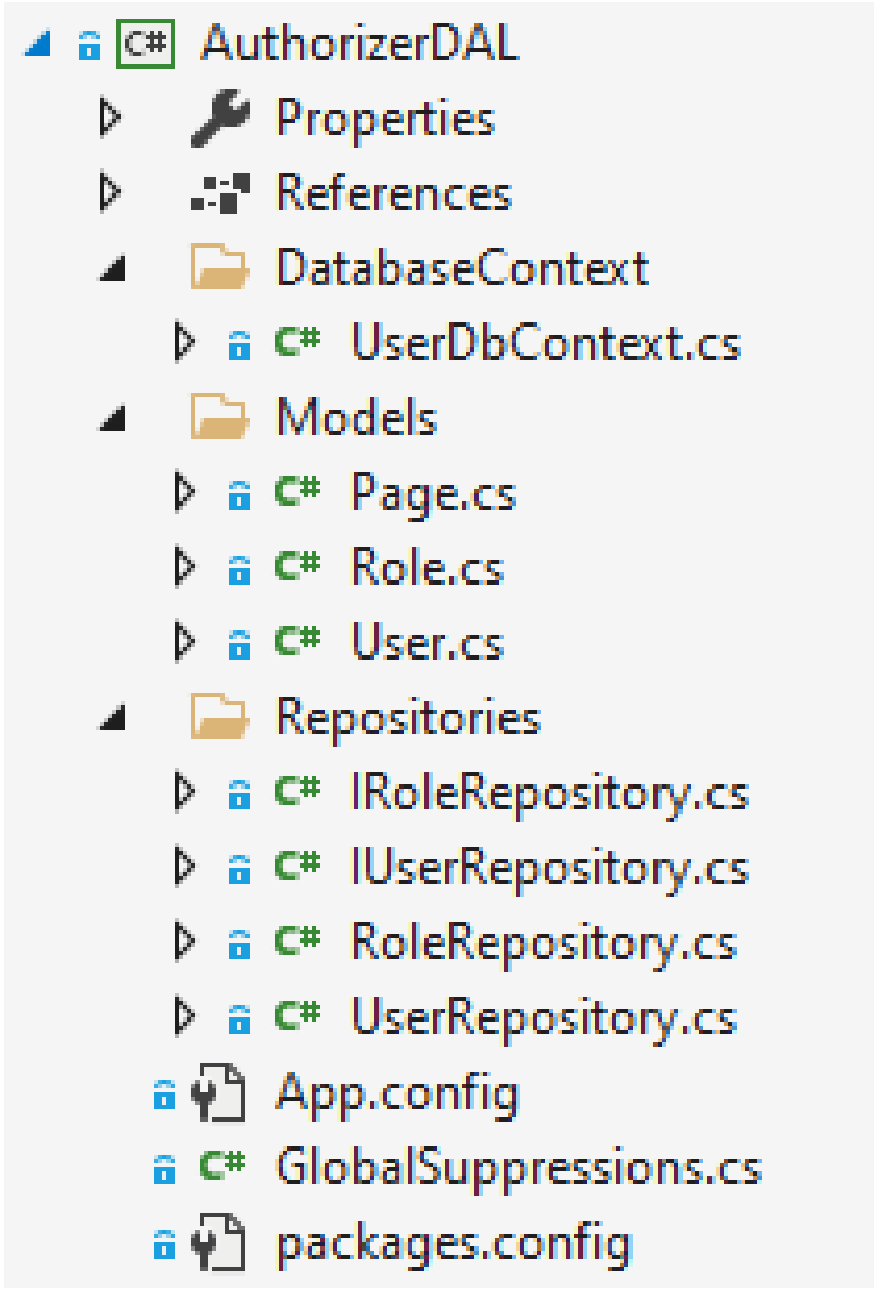
Пример


Solution 'AuthorizationProject' (4 of 4 projects)
AuthorizerBLL
AuthorizerDAL
AuthorizerPresentation
Common

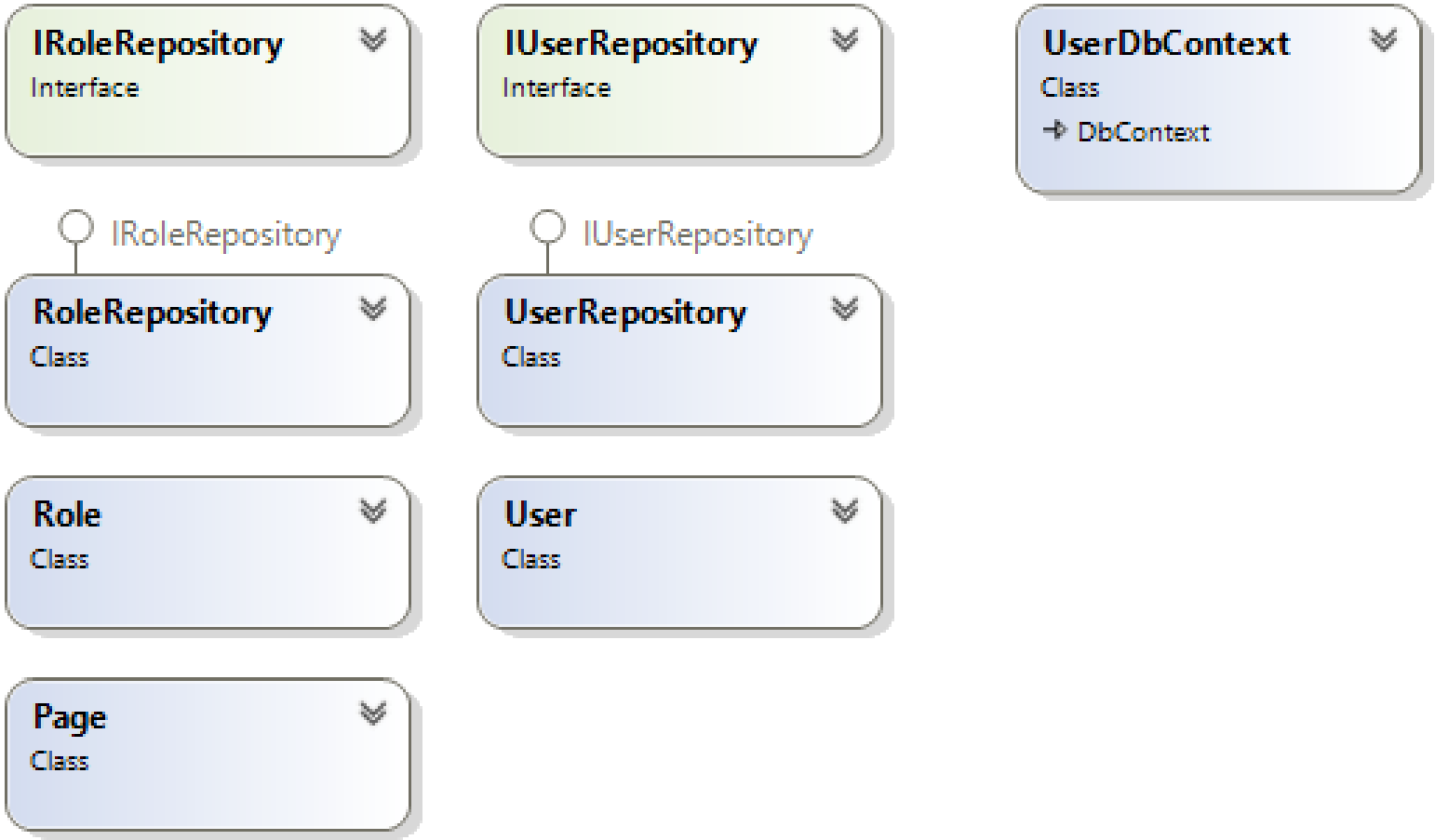
Пример


Web Application
AuthorizerPresentation
Dynamic Linked Library (DLL)
AuthorizerBLL
Dynamic Linked Library (DLL)
Common
Dynamic Linked Library (DLL)
AuthorizerDAL
Database

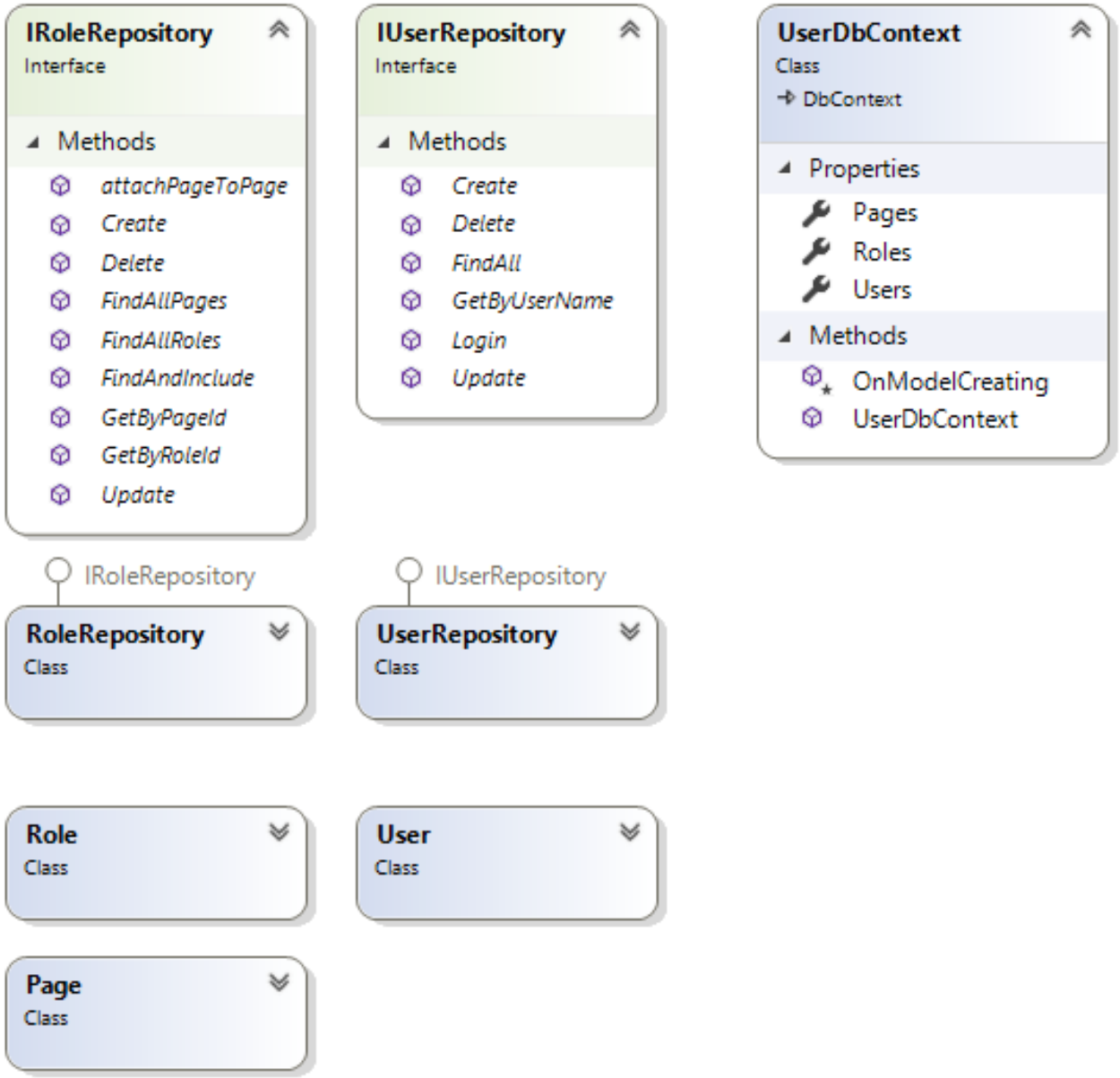
Пример


AuthorizerDAL
Properties
References
DatabaseContext
UserDbContext.cs
Models
Page.cs
Role.cs
User.cs
Repositories
IRoleRepository.cs
IUserRepository.cs
RoleRepository.cs
UserRepository.cs
App.config
GlobalSuppressions.cs
packages.config

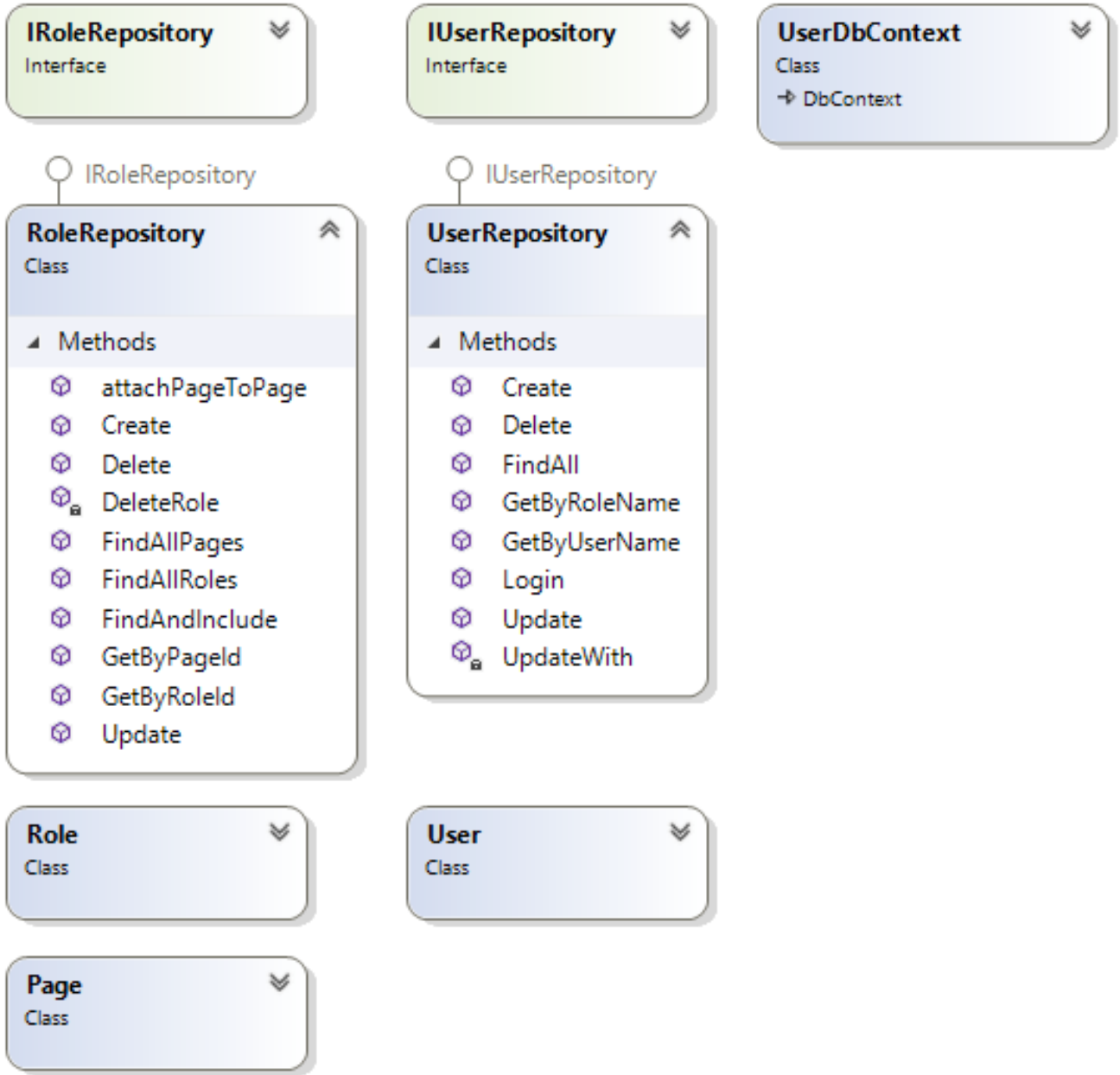
Пример


IRoleRepository
Interface
IUserRepository
Interface
UserDbContext
Class
DbContext
IRoleRepository
IUserRepository
RoleRepository
Class
UserRepository
Class
Role
Class
User
Class
Page
Class

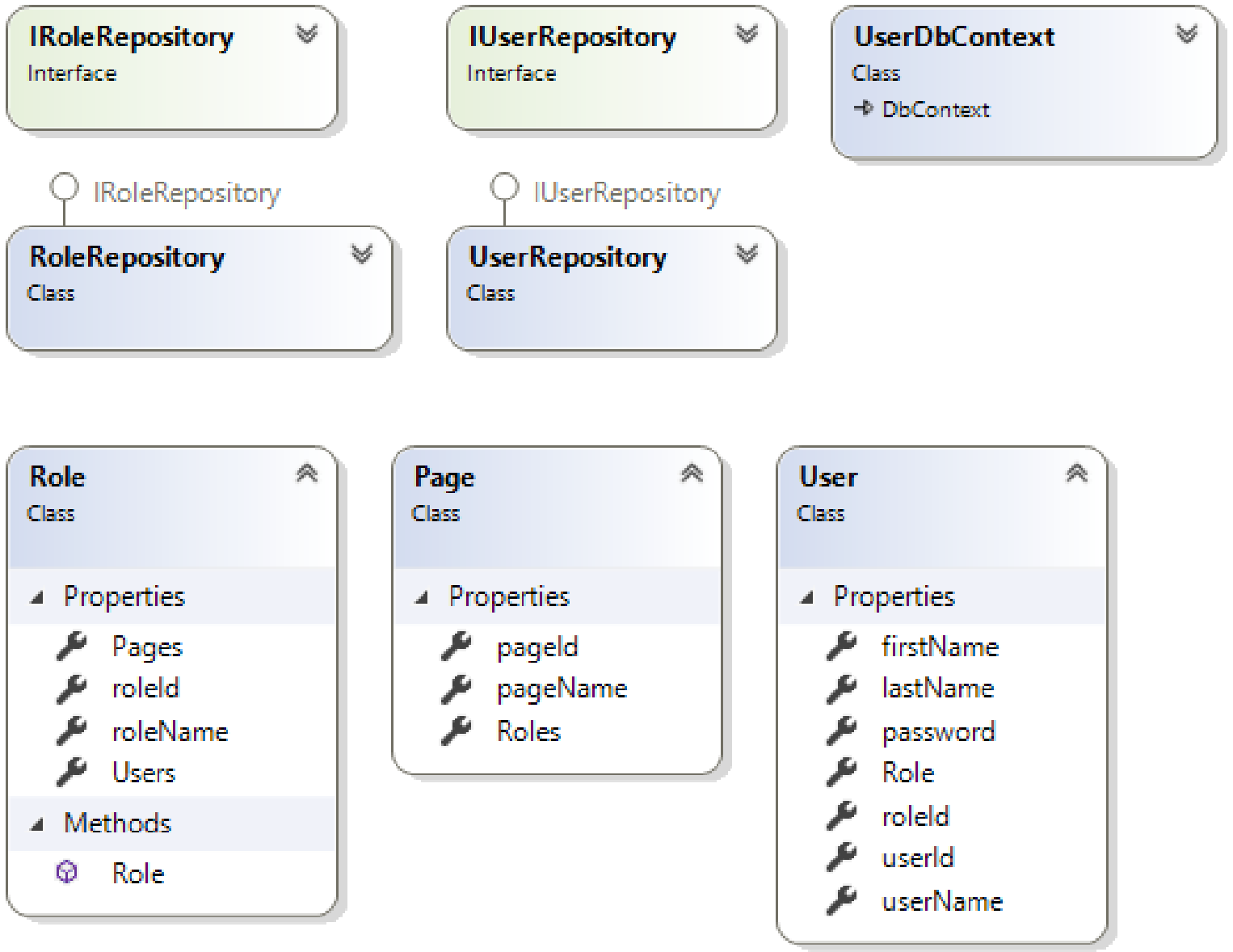
Пример


IRoleRepository
Interface
Methods
attachPageToPage
Create
Delete
FindAllPages
FindAllRoles
FindAndInclude
GetByPageId
GetByRoleId
Update
IUserRepository
Interface
Methods
Create
Delete
FindAll
GetByUserName
Login
Update
UserDbContext
Class
DbContext
Properties
Pages
Roles
Users
Methods
OnModelCreating
UserDbContext
IRoleRepository
IUserRepository
RoleRepository
Class
UserRepository
Class
Role
Class
User
Class
Page
Class

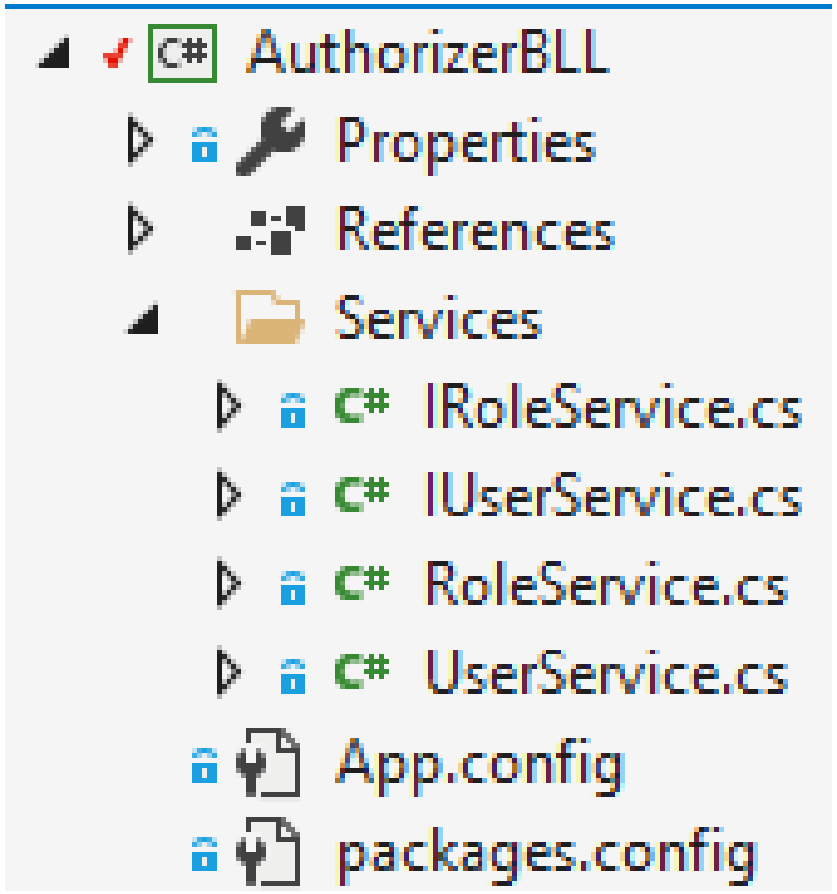
Пример


IRoleRepository
Interface
IUserRepository
Interface
UserDbContext
Class
DbContext
IRoleRepository
IUserRepository
RoleRepository
Class
Methods
attachPageToPage
Create
Delete
DeleteRole
FindAllPages
FindAllRoles
FindAndInclude
GetByPageId
GetByRoleId
Update
UserRepository
Class
Methods
Create
Delete
FindAll
GetByRoleName
GetByUserName
Login
Update
UpdateWith
Role
Class
User
Class
Page
Class

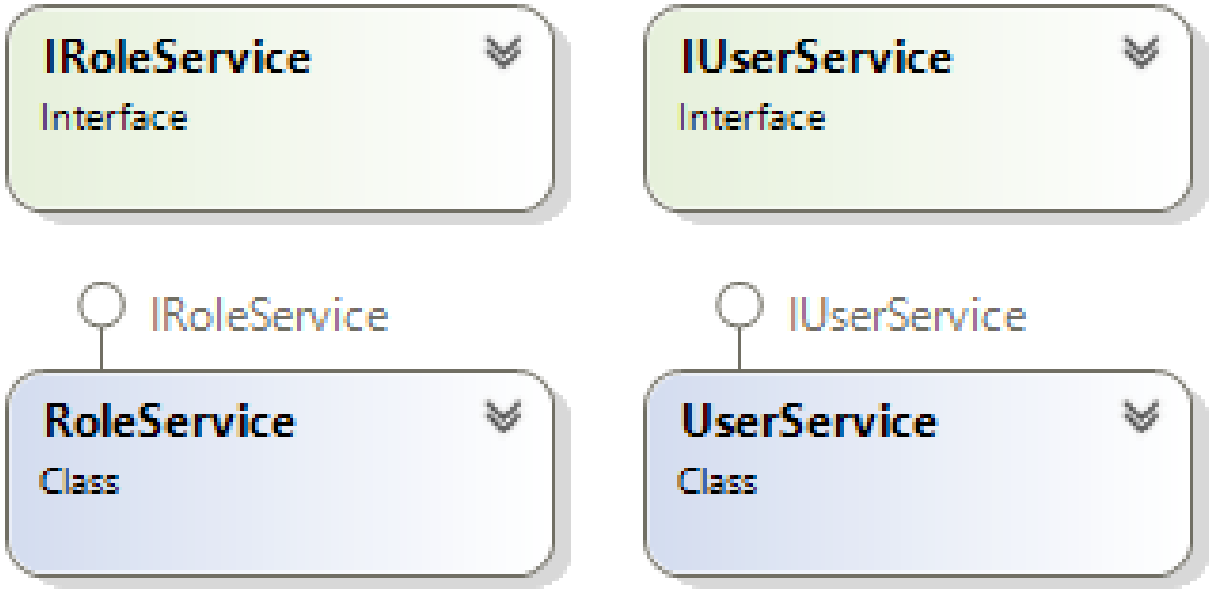
Пример


IRoleRepository
Interface
IUserRepository
Interface
UserDbContext
Class
DbContext
IRoleRepository
IUserRepository
RoleRepository
Class
UserRepository
Class
Role
Class
Properties
Pages
roleId
roleName
Users
Methods
Role
Page
Class
Properties
pageId
pageName
Roles
User
Class
Properties
firstName
lastName
password
Role
roleId
userId
userName

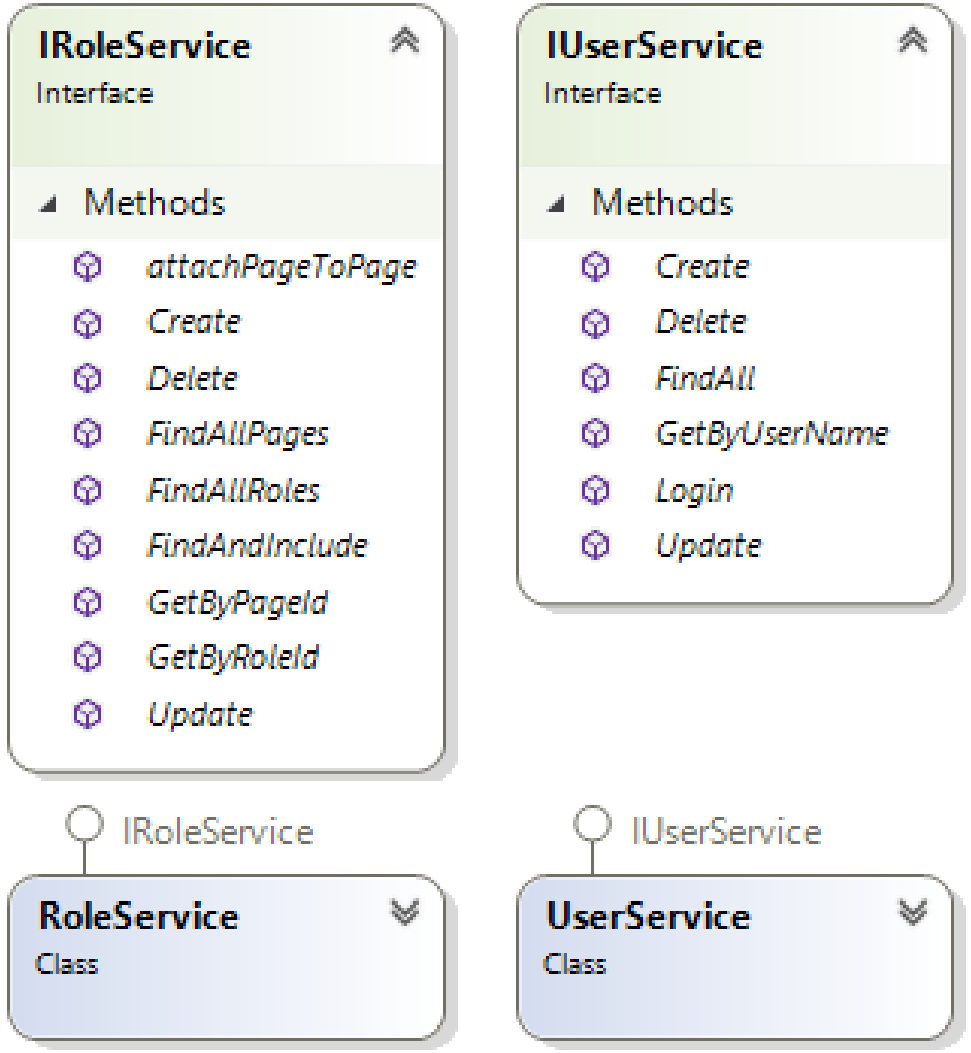
Пример


AuthorizerBLL
Properties
References
Services
IRoleService.cs
IUserService.cs
RoleService.cs
UserService.cs
App.config
packages.config

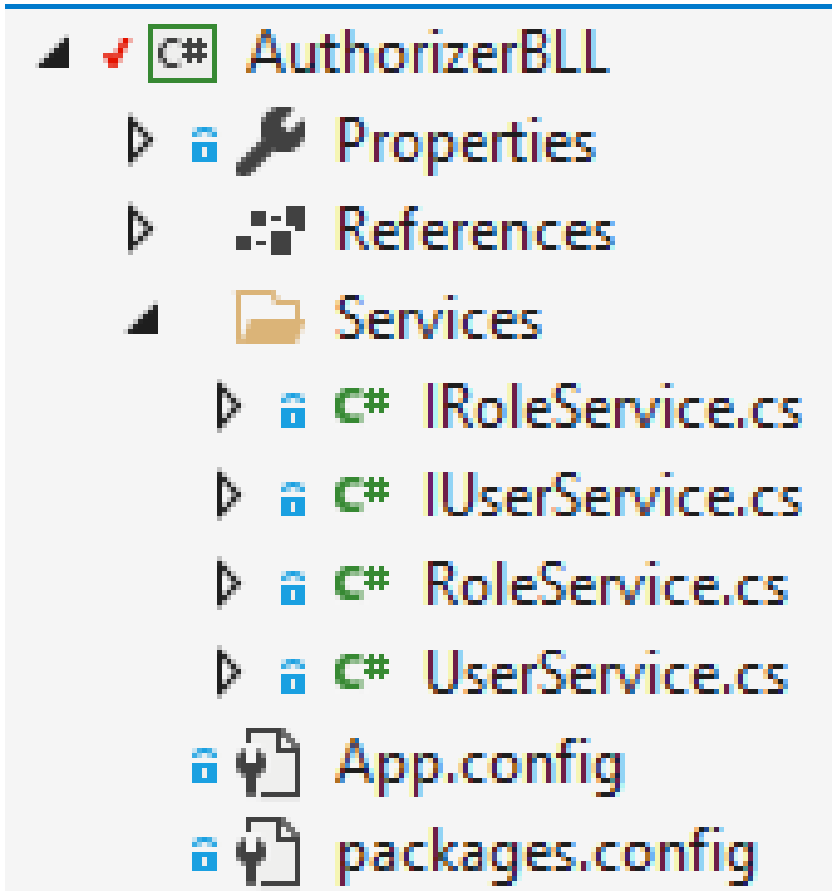
Пример


IRoleService
Interface
IUserService
Interface
IRoleService
IUserService
RoleService
Class
UserService
Class

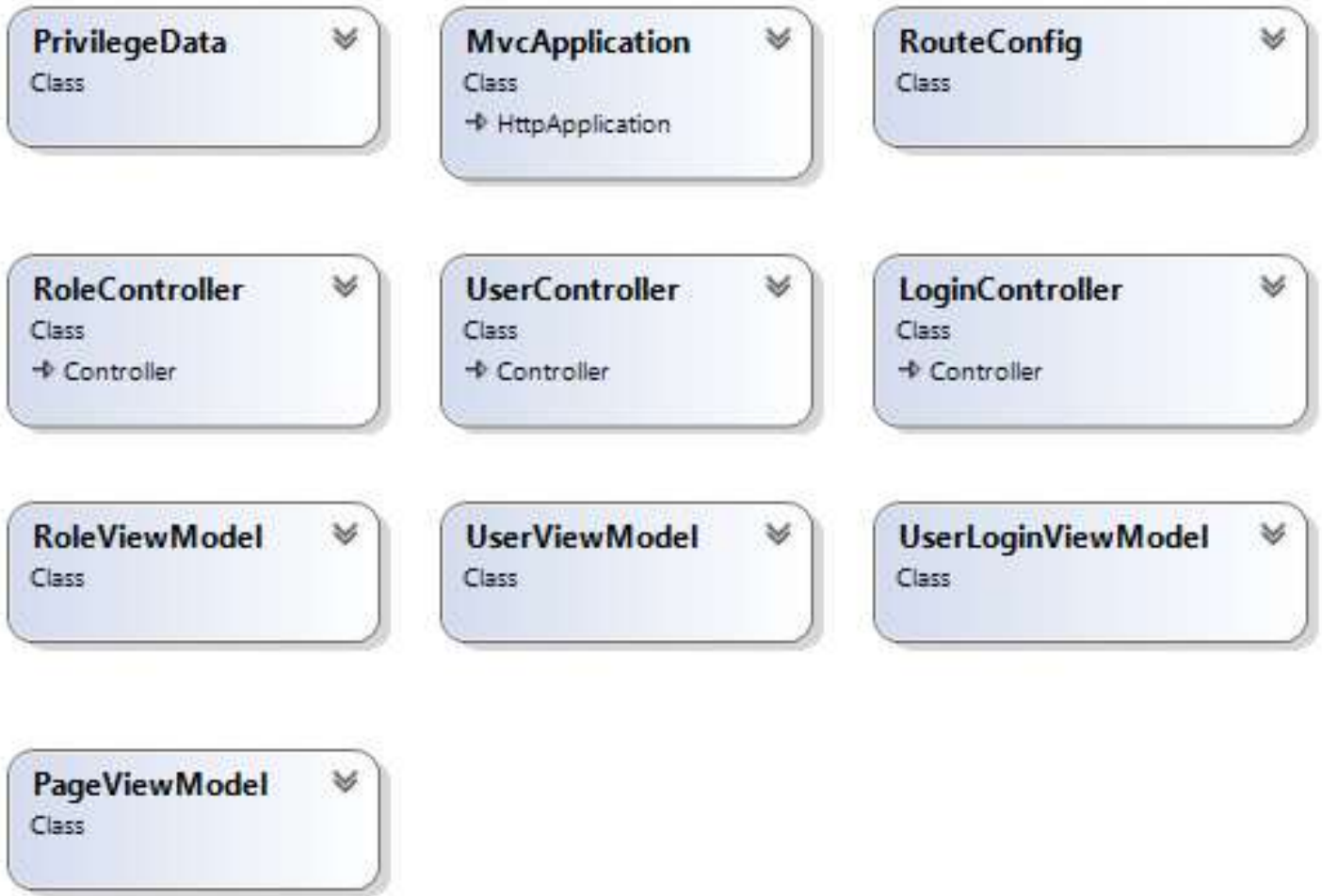
Пример


IRoleService
Interface
Methods
attachPageToPage
Create
Delete
FindAllPages
FindAllRoles
FindAndInclude
GetByPageId
GetByRoleId
Update
IUserService
Interface
Methods
Create
Delete
FindAll
GetByUserName
Login
Update
IRoleService
IUserService
RoleService
Class
UserService
Class

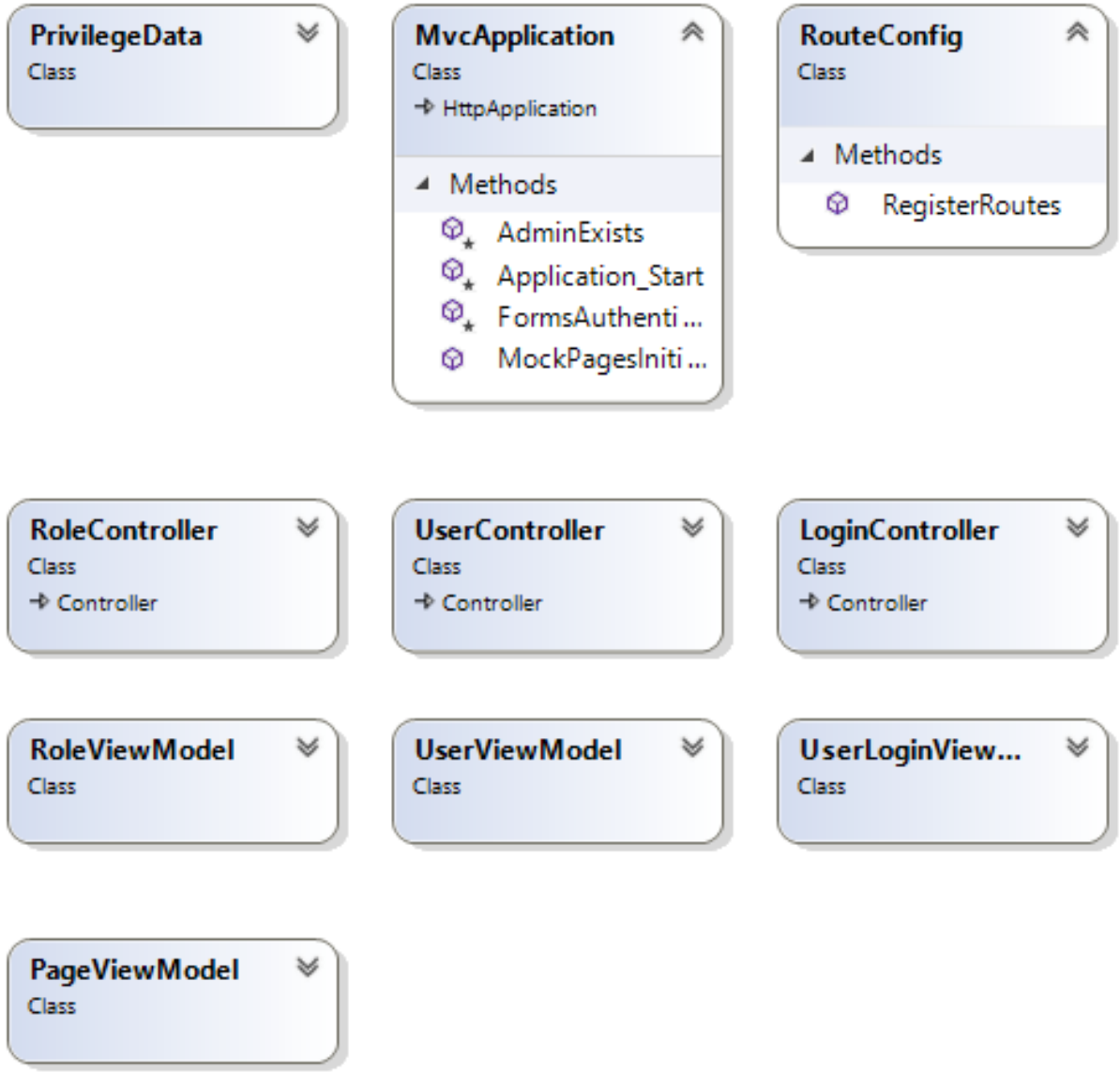
Пример


AuthorizerBLL
Properties
References
Services
IRoleService.cs
IUserService.cs
RoleService.cs
UserService.cs
App.config
packages.config

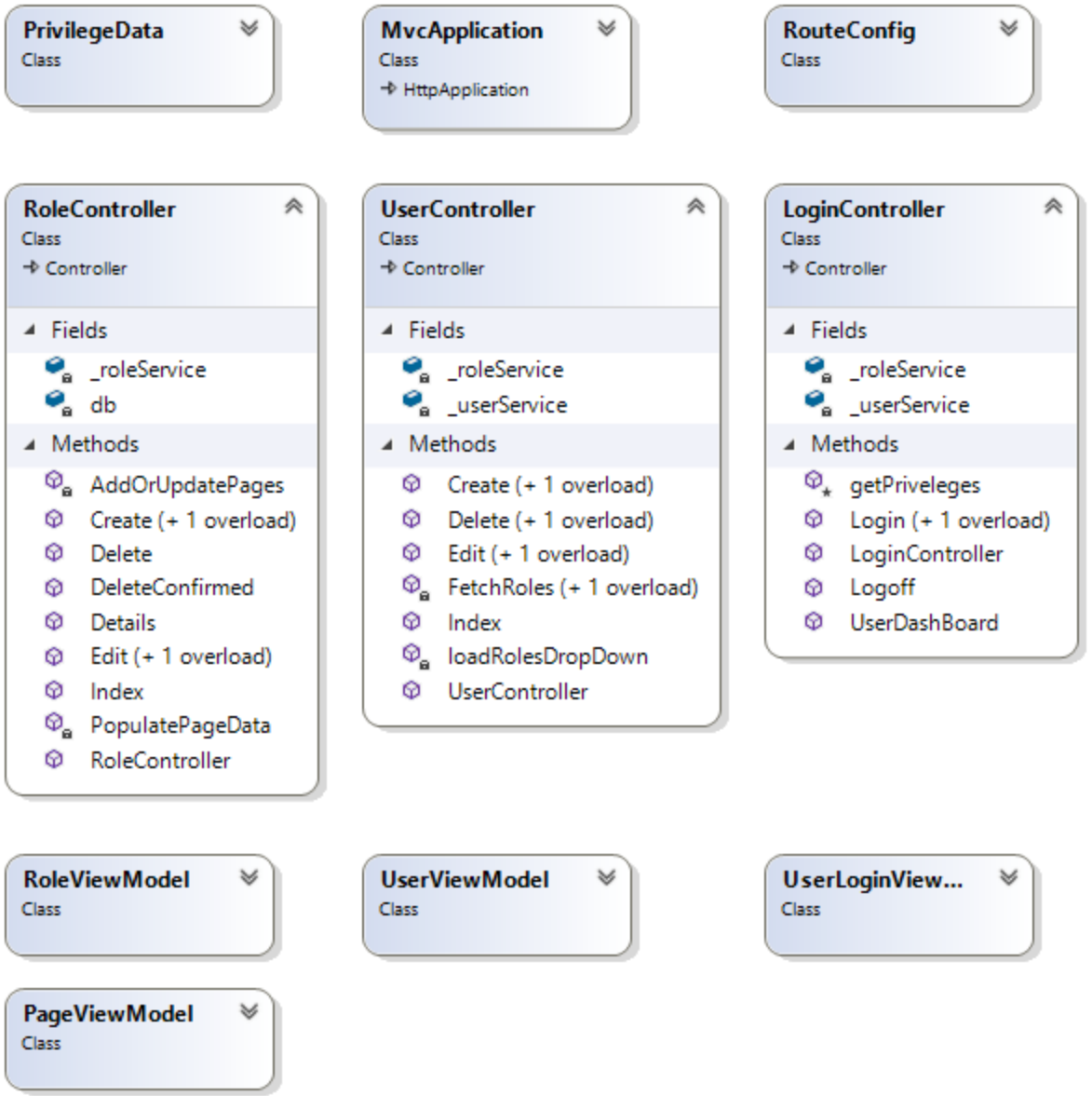
Пример


PrivilegeData
Class
MvcApplication
Class
HttpApplication
RouteConfig
Class
RoleController
Class
Controller
UserController
Class
Controller
LoginController
Class
Controller
RoleViewModel
Class
UserViewModel
Class
UserLoginViewModel
Class
PageViewModel
Class

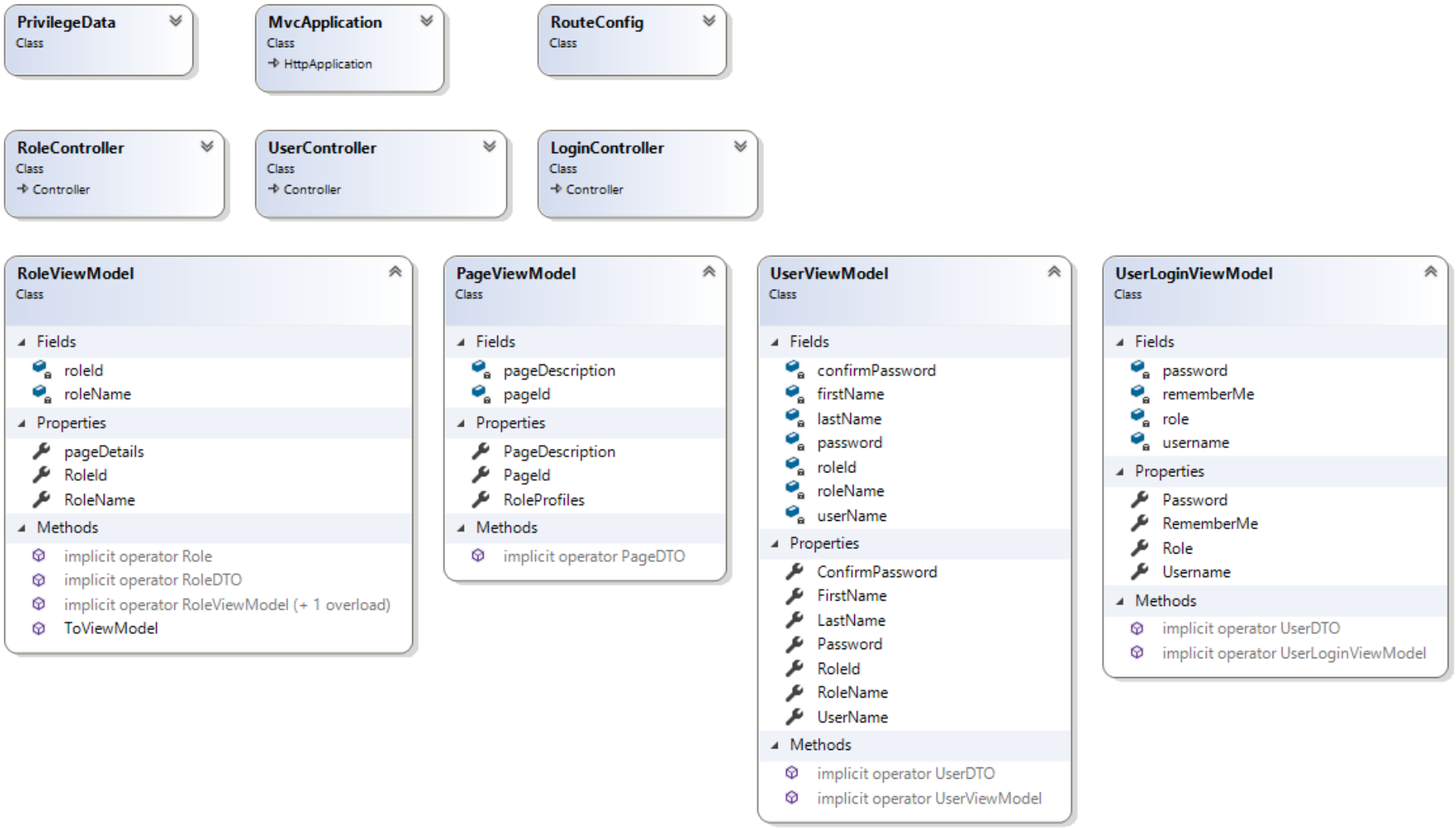
Пример


PrivilegeData
Class
MvcApplication
Class
HttpApplication
Methods
AdminExists
Application_Start
FormsAuthenti...
MockPagesIniti...
RouteConfig
Class
Methods
RegisterRoutes
RoleController
Class
Controller
UserController
Class
Controller
LoginController
Class
Controller
RoleViewModel
Class
UserViewModel
Class
UserLoginView...
Class
PageViewModel
Class

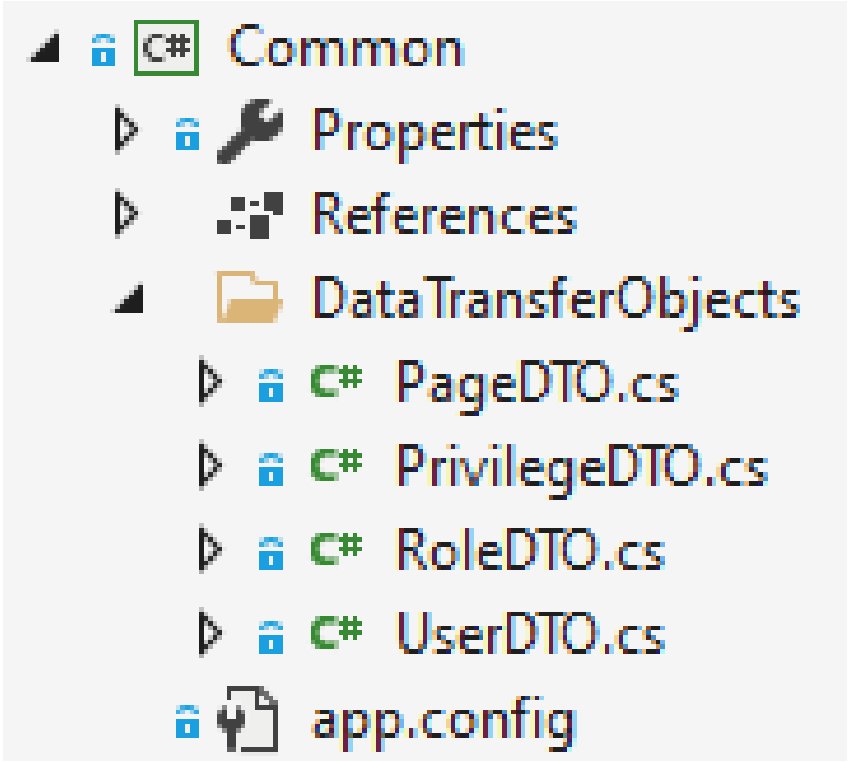
Пример


PrivilegeData
Class
MvcApplication
Class
HttpApplication
RouteConfig
Class
RoleController
Class
Controller
Fields
_roleService
db
Methods
AddOrUpdatePages
Create (+ 1 overload)
Delete
DeleteConfirmed
Details
Edit (+ 1 overload)
Index
PopulatePageData
RoleController
UserController
Class
Controller
Fields
_roleService
_userService
Methods
Create (+ 1 overload)
Delete (+ 1 overload)
Edit (+ 1 overload)
FetchRoles (+ 1 overload)
Index
loadRolesDropDown
UserController
LoginController
Class
Controller
Fields
_roleService
_userService
Methods
getPriveleges
Login (+ 1 overload)
LoginController
Logoff
UserDashBoard
RoleViewModel
Class
UserViewModel
Class
UserLoginView...
Class
PageViewModel
Class

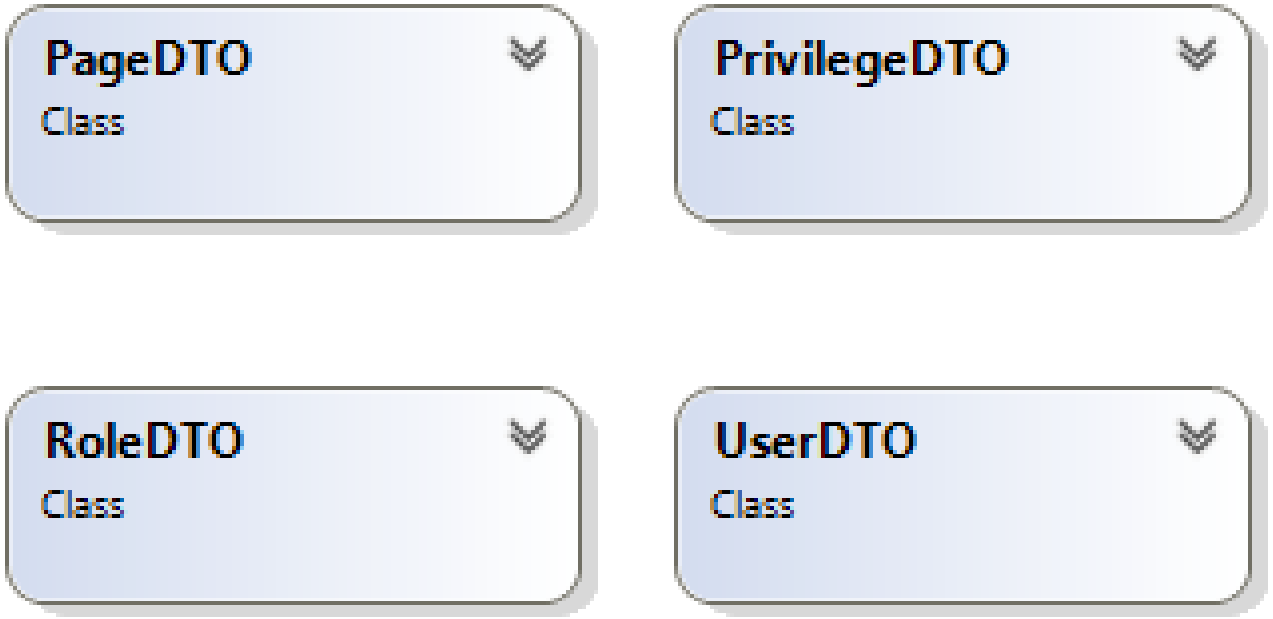
Пример


PrivilegeData
Class
MvcApplication
Class
HttpApplication
RouteConfig
Class
RoleController
Class
Controller
UserController
Class
Controller
LoginController
Class
Controller
RoleViewModel
Class
Fields
roleId
roleName
Properties
pageDetails
RoleId
RoleName
Methods
implicit operator Role
implicit operator RoleDTO
implicit operator RoleViewModel (+ 1 overload)
ToViewModel
PageViewModel
Class
Fields
pageDescription
pageId
Properties
PageDescription
PageId
RoleProfiles
Methods
implicit operator PageDTO
UserViewModel
Class
Fields
confirmPassword
firstName
lastName
password
roleId
roleName
userName
Properties
ConfirmPassword
FirstName
LastName
Password
RoleId
RoleName
UserName
Methods
implicit operator UserDTO
implicit operator UserViewModel
UserLoginViewModel
Class
Fields
password
rememberMe
role
username
Properties
Password
RememberMe
Role
Username
Methods
implicit operator UserDTO
implicit operator UserLoginViewModel

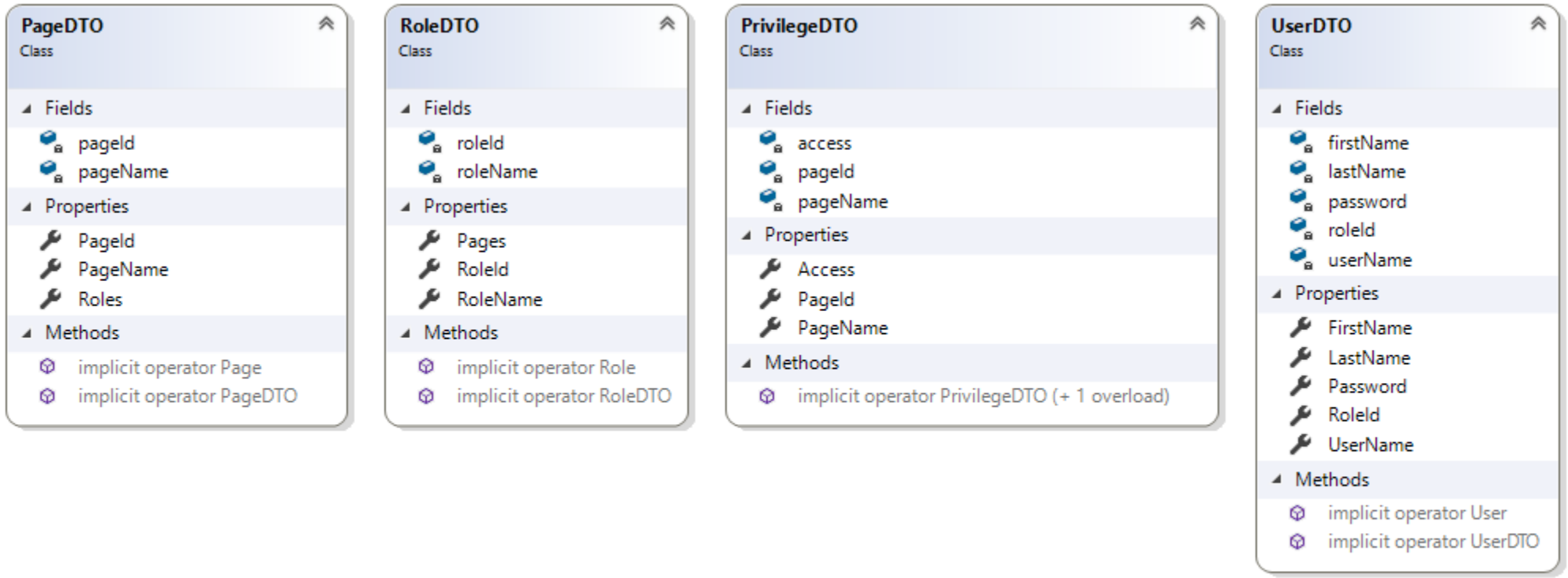
Пример


Common
Properties
References
DataTransferObjects
PageDTO.cs
PrivilegeDTO.cs
RoleDTO.cs
UserDTO.cs
app.config

Пример


PageDTO
Class
PrivilegeDTO
Class
RoleDTO
Class
UserDTO
Class

Пример


PageDTO
Class
Fields
pageId
pageName
Properties
PageId
PageName
Roles
Methods
implicit operator Page
implicit operator PageDTO
RoleDTO
Class
Fields
roleId
roleName
Properties
Pages
RoleId
RoleName
Methods
implicit operator Role
implicit operator RoleDTO
PrivilegeDTO
Class
Fields
access
pageId
pageName
Properties
Access
PageId
PageName
Methods
implicit operator PrivilegeDTO (+ 1 overload)
UserDTO
Class
Fields
firstName
lastName
password
roleId
userName
Properties
FirstName
LastName
Password
RoleId
UserName
Methods
implicit operator User
implicit operator UserDTO

Пример

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using AuthorizerDAL.DatabaseContext;
7  using AuthorizerDAL.Models;
8  using System.Data.Entity;
9  using System.Web.Mvc;
10
11 namespace AuthorizerDAL.Repositories
12 {
       1 reference
13     public class UserRepository : IUserRepository
14     {
           2 references
15         public int Create(User user)
16         {
17             try
18             {
19                 if (user != null)
20                 {
21                     using (var db = new UserDbContext())
22                     {
23                         db.Users.Add(user);
24                         db.SaveChanges();
25                     }
26                     return 1;
27                 }
28                 return 0;
29             }
30             catch(Exception)
31             {
32                 return -1;
33             }
34         }
```

Пример

```
2 references
public int Update(User user)
{
    if (user != null)
    {
        User userObj = GetByUserName(user.userName);
        userObj = UpdateWith(old: userObj, updated: user);
        using (var db = new UserDbContext())
        {
            db.Entry(userObj).State = EntityState.Modified;
            db.SaveChanges();
        }
        return 1;
    }
    return 0;
}

1 reference
private User UpdateWith(User old, User updated)
{
    old.firstName = updated.firstName;
    old.lastName = updated.lastName;
    old.roleId = updated.roleId;
    return old;
}
```

Пример

```
2 references
public int Delete(User user)
{
    if (user != null)
    {
        User userObj = GetByUserName(user.userName);
        using (var db = new UserDbContext())
        {
            db.Users.Attach(userObj);
            db.Users.Remove(userObj);
            db.SaveChanges();
        }
        return 1;
    }
    return 0;
}

2 references
public IList<User> FindAll()
{
    using (var db = new UserDbContext())
    {
        var userList = db.Users.ToList();
        return userList;
    }
}

4 references
public User GetByUserName(string userName)
{
    using (var db = new UserDbContext())
    {
        var a = db.Users.Where(u => u.userName == userName).FirstOrDefault();
        return a;
    }
}
```

Пример

```
        0 references
94      public Role GetByRoleName(string roleName)
95      {
96          using (var db = new UserDbContext())
97          {
98              var roleDetails = db.Roles.Where(r => r.roleName == roleName).FirstOrDefault();
99              return roleDetails;
100         }
101     }
102
        2 references
103     public User Login(User loginUser)
104     {
105         using (var db = new UserDbContext())
106         {
107             var obj = db.Users.Where(u => u.userName.Equals(loginUser.userName) && u.password.Equals(loginUser.password)).FirstOrDefault();
108             if (obj != null)
109             {
110                 return obj;
111             }
112             else
113                 return null;
114         }
115     }
116 }
117 }
118
```

Пример

```
10  namespace AuthorizerBLL.Services
11  {
12      public class UserService : IUserService
13      {
14
15          private readonly IUserRepository _iUserRepository;
16
17          //DI to UserRepository
            0 references
18          public UserService(IUserRepository userRepository)
19          {
20              _iUserRepository = userRepository;
21          }
22
23          public int Create(UserDTO user)
24          {
25              return _iUserRepository.Create(user);
26          }
27
28
29          public int Update(UserDTO user)
30          {
31              return _iUserRepository.Update(user);
32          }
33
34          public int Delete(UserDTO user)
35          {
36              return _iUserRepository.Delete(user);
37          }
38
```

Пример

```
2 references
39  public IList<UserDTO> FindAll()
40  {
41      return _iUserRepository.FindAll().Select(b => new UserDTO()
42      {
43          FirstName = b.firstName,
44          LastName = b.lastName,
45          Password = b.password,
46          UserName = b.userName,
47          RoleId = b.roleId
48      }).ToList();
49  }
50
    3 references
51  public UserDTO GetByUserName(string userName)
52  {
53      if (userName == null)
54      {
55          return null;
56      }
57      else
58      {
59          //Removes empty spaces if any
60          var checkUserName = userName.Trim();
61          return _iUserRepository.GetByUserName(userName);
62      }
63  }
```

Пример

```
2 references
public UserDTO Login(UserDTO loginUser)
{
    var loginResult = _iUserRepository.Login(loginUser);
    if (loginResult!=null)
    {
        return loginResult;
    }
    else
    {
        return null;
    }
}
}
}
```

Пример

```
 8  namespace AuthorizerPresentation.Controllers
 9  {
10      [Authorize(Roles = "admin,superuser")]
        1 reference
11      public class UserController : Controller
12      {
13          private readonly IUserService _userService;
14          private readonly IRoleService _roleService;
15
            0 references
16          public UserController(IUserService userService, IRoleService roleService)
17          {
18              _userService = userService;
19              _roleService = roleService;
20          }
21
```

Пример

```
22      /// <summary>
23      /// Display list of users
24      /// </summary>
25      /// <returns></returns>
        0 references
26      public ActionResult Index()
27      {
28          var userDtoList = _userService.FindAll();
29
30          if (userDtoList != null)
31          {
32              //Mapping UserDto to UserViewModel
33              IList<UserViewModel> users = userDtoList.Select(b => new UserViewModel()
34              {
35                  FirstName = b.FirstName,
36                  LastName = b.LastName,
37                  Password = b.Password,
38                  UserName = b.UserName,
39                  RoleId = b.RoleId,
40                  RoleName = FetchRoles(b.RoleId)
41              }).ToList();
42              return View(users);
43          }
44          else
45          {
46              return View();
47          }
48
49      }
```

Пример

```
51      //Create a new user
        0 references
52      public ActionResult Create()
53      {
54          loadRolesDropDown();
55          return View();
56      }
57      [HttpPost]
        0 references
58      public ActionResult Create(UserViewModel userView)
59      {
60          if (userView != null)
61          {
62              loadRolesDropDown();
63              userView.FirstName.Trim();
64              var userCreateResult = _userService.Create(userView);
65
66              if (userCreateResult == -1)
67              {
68                  ModelState.AddModelError(key: "UserName", errorMessage: "User Name Already Exists");
69                  return View(userView);
70              }
71              else if (userCreateResult == 0)
72              {
73                  ViewBag.Message = "Db Creation Error! Please Restart Application";
74              }
75              return RedirectToAction("Index");
76          }
77          else
78          {
79              return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
80          }
81      }
```

Пример

```
83          //Edit existing user
            0 references
84          public ActionResult Edit(string username)
85          {
86              if (username == null)
87              {
88                  return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
89              }
90              UserViewModel userCollection = _userService.GetByUserName(username);
91              if(userCollection == null)
92              {
93                  return HttpNotFound();
94              }
95              loadRolesDropDown();
96              return View(userCollection);
97          }
98
99          [HttpPost]
100         [ValidateAntiForgeryToken]
            0 references
101         public ActionResult Edit(UserViewModel userDetails, FormCollection requestValidator)
102         {
103             _userService.Update(userDetails);
104             return RedirectToAction("Index");
105         }
106
107         //Delete a user
            0 references
108         public ActionResult Delete(string username)
109         {
110             if (username == null)
111             {
112                 return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
113             }
114             else
115             {
116                 UserViewModel userCollection = _userService.GetByUserName(username);
117                 if (HttpContext.User.Identity.Name == username)
118                 {
119                     ViewBag.Error = "Cannot delete yourself";
120                     return View();
121                 }
122                 else
123                 {
124                     return View(userCollection);
125                 }
126             }
127         }
```

Пример

```
128         [HttpPost]
129         [ValidateAntiForgeryToken]
            0 references
130         public ActionResult Delete(UserViewModel userDetails, FormCollection requestValidator)
131         {
132             if (userDetails == null)
133             {
134                 return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
135             }
136             else
137             {
138                 _userService.Delete(userDetails);
139                 return RedirectToAction("Index");
140             }
141         }
142
143         /// <summary>
144         /// Fetches list of all roles from db
145         /// </summary>
            2 references
146         private SelectList FetchRoles()
147         {
148             ViewData.Clear();
149
150             //Obtaining list of roles
151             var roleList = _roleService.FindAllRoles().ToList();
152
153             //Setting to select list for populating combo box
154             return new SelectList(roleList, dataValueField: "roleId", dataTextField: "roleName");
155         }
```

Пример

```
157         /// <summary>
158         /// Fetches role based on id
159         /// </summary>
160         /// <param name="id"></param>
161         /// <returns></returns>
            1 reference
162         private string FetchRoles(int id)
163         {
164             ViewData.Clear();
165
166             //Obtaining list of roles
167             var roleList = _roleService.FindAllRoles().ToList();
168
169             return roleList.Where(r => r.RoleId == id).Select(r => r.RoleName).First();
170         }
171
172         /// <summary>
173         /// Sets List of roles to limit showing superuser unless user is superuser
174         /// </summary>
            3 references
175         private void loadRolesDropDown()
176         {
177             if (!User.IsInRole("superuser"))
178             {
179                 ViewData["Roles"] = new SelectList(items: FetchRoles().Where(r => r.Text != "superuser").ToList(), dataValueField: "Value", dataTextField: "Text");
180             }
181             else
182             {
183                 ViewData["Roles"] = FetchRoles();
184             }
185         }
186     }
187 }
```

Спасибо за внимание!