1) Класс DbUser представляет собой модель пользователя в базе данных. Он содержит свойства, которые соответствуют полям таблицы пользователей в базе данных.

Свойства класса:

- IdUserRole: идентификатор роли пользователя;

- Login: логин пользователя;

- HashPassword: хэш пароля пользователя;

- FirstName: имя пользователя;

- SecondName: фамилия пользователя;

- DateBirthday: дата рождения пользователя;

- IsBlocked: флаг блокировки пользователя.

Конструкторы:

- DbUser(): конструктор по умолчанию;

- DbUser(uint idUserRole, string login, byte[] hashPassword, string firstName, string secondName,

DateTime dateBirthday, bool isBlocked): конструктор с параметрами, позволяющий задать значения свойств объекта при создании.

Методы:

- ToString(): переопределенный метод, который возвращает строковое представление объекта класса DbUser.

2) Класс HashedValue представляет собой модель хэшированного значения. Он содержит свойства, которые позволяют задать и получить оригинальное значение и его хэш-значение.

Свойства класса:

- OriginalValue: оригинальное значение;

- Value: хэш-значение.

Конструкторы:

- HashedValue(): конструктор по умолчанию;

- HashedValue(string originalValue): конструктор с параметром, позволяющий задать оригинальное значение при создании объекта.

Методы:

- ToString(): переопределенный метод, который возвращает строковое представление объекта класса HashedValue.

Класс использует метод ConvertStringToHeshString из библиотеки Xk7.Helper.Converts для хэширования значения.

3) Класс Slot представляет модель слота в расписании. Он содержит свойства для идентификатора слота, логина сотрудника, даты и времени слота. Конструктор класса позволяет создавать объекты Slot с заданными значениями свойств.

Свойство Id представляет идентификатор слота. Свойство EmployeeLogin содержит логин сотрудника, на которого назначен слот. Свойство SlotDate содержит дату слота, а свойство SlotTime - время слота.

Конструктор класса принимает параметры idTimeTable, employeeLogin, slotDate и slotTime, которые задают соответствующие свойства объекта.

Класс Slot реализует интерфейс ISlot, который определяет свойства, необходимые для работы со слотами в расписании.

4) Статический класс DbSettingsService предоставляет методы для работы с файлом настроек базы данных.

Метод LoadDbSettings загружает настройки подключения к базе данных из файла SettingsMySql.json и возвращает объект DbConnectionStringBuilder, содержащий эти настройки.

Метод DbSettingsFileExists проверяет, существует ли файл настроек базы данных.

Метод RemoveDbSettings удаляет файл настроек базы данных.

Все методы класса являются статическими, что позволяет использовать их без создания экземпляра класса. Класс использует библиотеку Newtonsoft.Json для десериализации настроек из файла в объект типа MySqlConnectionStringBuilder

5) Статический класс UICultureService предоставляет методы для работы с культурами приложения.

Метод GetAllCultures возвращает список всех доступных культур приложения, отфильтрованных по текущей культуре пользователя.

Метод GetProperty возвращает строковое значение свойства ресурса по его имени из файла ресурсов приложения.

Метод GetProperty с параметром cultureInfo возвращает строковое значение свойства ресурса на определенной культуре из файла ресурсов приложения.

Метод SetCulture устанавливает текущую культуру приложения.

Метод GetDefaultCulture возвращает текущую культуру приложения по умолчанию.

Класс использует файл ресурсов приложения для получения строковых значений свойств.

6) Класс LoginViewModel отвечает за логику авторизации пользователя в приложении. Он содержит поля для ввода имени пользователя и пароля, а также поле для вывода сообщений об ошибках. Класс также содержит команды для выполнения процесса авторизации и перехода к регистрации нового пользователя.

Поля класса:

- Username - строковое поле для ввода имени пользователя.

- Password - поле типа SecureString для ввода пароля.

- ErrorMessage - строковое поле для вывода сообщений об ошибках.

- isViewVisible - булевое поле, которое определяет видимость представления.

Команды класса:

- LoginCommand - команда для выполнения процесса авторизации.

- LoginEmployeeCommand - команда для выполнения авторизации сотрудника.

- RegistrationCommand - команда для перехода к регистрации нового пользователя.

Методы класса:

- CanExecuteLoginCommand - метод, который проверяет возможность выполнения команды LoginCommand.

- ExecuteLoginCommand - метод, который осуществляет процесс авторизации пользователя.

Класс использует интерфейс ICommand для выполнения команд и наследует класс ViewModelBase, который реализует интерфейс INotifyPropertyChanged для обновления данных в представлении.

7) Класс ViewModelBase является базовым классом для всех ViewModel в приложении. Он реализует интерфейс INotifyPropertyChanged для обновления данных в представлении. Класс содержит событие PropertyChanged, которое вызывается при изменении свойств объекта. Класс также содержит метод OnPropertyChanged, который вызывает событие PropertyChanged и передает имя измененного свойства в качестве параметра.

Поля класса:

- PropertyChanged - событие, которое вызывается при изменении свойств объекта.

Методы класса:

- OnPropertyChanged - метод, который вызывает событие PropertyChanged и передает имя измененного свойства в качестве параметра.

8) Класс ViewModelCommand реализует интерфейс ICommand для создания команд, которые могут быть связаны с элементами управления в представлении. Класс содержит два конструктора, которые принимают делегаты для выполнения команды и проверки ее возможности выполнения.

Поля класса:

- \_executeAction - делегат, который содержит метод для выполнения команды.

- \_canExecutePredicate - делегат, который содержит метод для проверки возможности выполнения команды.

Методы класса:

- CanExecute - метод, который проверяет возможность выполнения команды и возвращает результат проверки.

- Execute - метод, который выполняет команду.

- CanExecuteChanged - событие, которое вызывается при изменении возможности выполнения команды и регистрируется в CommandManager.

9) Определение класса AdminPanel наследуется от класса Page. Он представляет собой страницу приложения для администратора. В конструкторе определяется сервис для работы с базой данных и проверяется его наличие.

Метод CreateAsync используется для создания экземпляра класса AdminPanel асинхронно. Он получает данные о пользователях из базы данных и заполняет ими таблицу на странице.

Метод ChangeLanguageClick вызывается при нажатии на кнопку смены языка. Он меняет языковую культуру приложения на английскую, если текущая языковая культура является русской, и на русскую в противном случае.

Метод ExitClick вызывается при нажатии на кнопку выхода из страницы. Он перенаправляет пользователя на страницу авторизации.

Метод AdminPanelEditButton вызывается при нажатии на кнопку редактирования данных пользователя. Он получает выбранного пользователя из таблицы и передает его в качестве параметра на страницу редактирования пользователей.

10) Класс EditUser представляет страницу редактирования данных пользователя. В конструктор класса передается сервис IDbAsyncService и объект DbUser, который будет редактироваться. На странице расположены элементы управления для ввода новых данных пользователя, а также кнопки подтверждения и отмены изменений.

Метод ConfirmButton\_Click обрабатывает нажатие кнопки подтверждения изменений. В методе создается копия объекта DbUser, который будет редактироваться, и заполняется новыми данными из элементов управления на странице. Затем вызывается метод UpdateUserTableAsync сервиса IDbAsyncService для сохранения изменений в базе данных. После этого происходит перенаправление пользователя на страницу AdminPanel.

Метод CancelButton\_Click обрабатывает нажатие кнопки отмены изменений. В методе происходит перенаправление пользователя на страницу AdminPanel без сохранения изменений.

Оба метода используют метод CreateAsync класса AdminPanel для создания экземпляра страницы AdminPanel с актуальными данными из базы данных.

11) Класс UserProfile представляет страницу редактирования данных пользователя. В конструктор класса передается сервис IDbAsyncService и объект DbUser, который будет редактироваться. На странице расположены элементы управления для ввода новых данных пользователя, а также кнопки подтверждения и отмены изменений.

Метод ConfirmButton\_Click обрабатывает нажатие кнопки подтверждения изменений. В методе создается копия объекта DbUser, который будет редактироваться, и заполняется новыми данными из элементов управления на странице. Затем вызывается метод UpdateUserTableAsync сервиса IDbAsyncService для сохранения изменений в базе данных. После этого происходит перенаправление пользователя на страницу AdminPanel.

Метод CancelButton\_Click обрабатывает нажатие кнопки отмены изменений. В методе происходит перенаправление пользователя на страницу AdminPanel без сохранения изменений.

Оба метода используют метод CreateAsync класса AdminPanel для создания экземпляра страницы AdminPanel с актуальными данными из базы данных. Кроме того, класс содержит методы для изменения языка приложения и выхода из аккаунта.

12) Класс App представляет главное приложение. В конструкторе класса создается объект NavigationWindow, который будет использоваться для навигации между страницами приложения. Также в конструкторе класса устанавливается текущая культура приложения и создается объект сервиса для работы с базой данных.

Метод FatalError используется для обработки фатальных ошибок приложения. В случае возникновения ошибки метод отображает сообщение об ошибке и завершает работу приложения.

Метод ConfigureDefaultDbService используется для создания объекта сервиса для работы с базой данных. В методе проверяется наличие файла настроек базы данных и загружаются настройки из файла. Затем создается объект сервиса с использованием настроек из файла.

Класс также содержит методы для изменения языка приложения и выхода из аккаунта.

13) Класс AppEnvironment содержит статический метод GetRootWorkDirectory, который возвращает путь к корневой директории проекта.

Метод использует метод GetCurrentDirectory класса Directory для получения текущей директории приложения. Затем в цикле while метод проверяет, существует ли директория "bin" в текущей директории. Если директория не найдена, метод переходит к родительской директории и продолжает поиск.

Если директория "bin" найдена, метод возвращает путь к корневой директории проекта. Если же директория "bin" не найдена, метод возвращает пустую строку.

14) Класс UserFactory содержит статический метод FromDataRow, который преобразует объект типа DataRow в объект типа, реализующий интерфейс IDbUser.

Метод использует обобщенный тип T для создания нового объекта типа T и заполнения его свойствами из переданного DataRow.

Если тип T не является наследником интерфейса IDbUser, метод генерирует исключение FactoryException.

Метод SetItemFromRow используется для заполнения свойств объекта из DataRow. Если значение в DataRow равно DBNull.Value, свойство объекта остается неизменным.

15) Класс Converts содержит два статических метода для преобразования данных.

Метод ConvertByteArrayToString принимает массив байтов и возвращает строку, созданную из него с помощью кодировки UTF-8.

Метод ConvertStringToHeshString принимает строку и возвращает ее хеш-значение в виде строки, созданное с помощью алгоритма MD5. Для этого метод создает объект класса MD5 из пространства имен System.Security.Cryptography, преобразует входную строку в массив байтов с помощью кодировки ASCII, вычисляет хеш-значение этого массива и возвращает его в виде строки, созданной с помощью метода Convert.ToHexString.

Методы могут быть полезны для работы с данными, хранящимися в бинарном или строковом формате.

16) Определение перечисления AddSlotResult содержит четыре возможных значения результата добавления слота.

Значение Success указывает на успешное добавление слота.

Значение NotExistsUser указывает на то, что пользователь не существует.

Значение WrongUserRole указывает на неправильную роль пользователя.

Значение Unknown указывает на неизвестную ошибку при добавлении слота.

17) Класс User определяет свойства и методы для пользователя в приложении.

Класс содержит следующие свойства:

IdUserRole: перечисление, которое представляет роль пользователя (например, администратор, редактор, гость и т.д.).

Login: строка, которая представляет имя пользователя для входа в систему.

Password: зашифрованное значение пароля пользователя.

FirstName: строка, которая представляет имя пользователя.

SecondName: строка, которая представляет фамилию пользователя.

DateBirthday: дата рождения пользователя в формате DateOnly.

IsBanned: булево значение, которое указывает, заблокирован ли пользователь в системе. Значение по умолчанию - false.

Класс имеет два конструктора: один без параметров и один, который принимает значения для всех свойств. Также в классе есть переопределенный метод ToString(), который возвращает строку с текущими значениями всех свойств объекта класса User.

18) Класс DbAsyncService содержит методы для работы с базой данных. Класс реализует интерфейс IDbAsyncService и содержит следующие методы:

ExistsUserAsync(string login) - метод для проверки существования пользователя в базе данных по логину.

IsBannedUserAsync(string login) - метод для проверки, заблокирован ли пользователь.

GetHashPasswordAsync(string login) - метод для получения хэша пароля пользователя.

AddUserAsync(User user) - метод для добавления нового пользователя в базу данных.

GetDataUserByLoginAsync(string login) - метод для получения данных пользователя по логину.

AddLog(string login, LoggingType loggingType) - метод для добавления записи в журнал.

UpdateUserTableAsync(string OldLogin, DbUser NewUser) - метод для обновления данных пользователя в базе данных.

Класс использует асинхронный подход для взаимодействия с базой данных. Внутри методов происходит проверка состояния подключения к базе данных и выброс соответствующих исключений в случае проблем с подключением или выполнением запроса.

19) Класс DbService реализует интерфейс IDbService. У класса есть несколько методов, которые выполняют операции с базой данных, такие как проверка существования пользователя, добавление пользователя, получение пользовательских данных, добавление журнала, получение таблицы и обновление пользовательской таблицы.

Конструктор инициализирует класс с помощью объекта DbConnection и открывает соединение. Если соединение не открывается, генерируется исключение ConnectionException.

Метод ExistsUser проверяет, существует ли пользователь в базе данных, выполняя запрос SQL. Если пользователь существует, возвращается true; в противном случае возвращается ложь.

Метод IsBannedUser проверяет, заблокирован ли пользователь, выполняя SQL-запрос. Если пользователь забанен, возвращается true; в противном случае возвращается ложь.

Метод GetHashPassword извлекает хэш-пароль пользователя из базы данных, выполняя запрос SQL.

Метод AddUser добавляет нового пользователя в базу данных, выполняя SQL-запрос INSERT. Если пользователь уже существует, возвращается AddUserResult.UserExists; если операция выполнена успешно, возвращается AddUserResult.Success; в противном случае возвращается AddUserResult.Unknown.

Метод GetDataUserByLogin извлекает все данные для пользователя из базы данных, выполняя запрос SQL, и возвращает их в виде объекта DataRow.

Метод AddLog добавляет новый журнал в базу данных, выполняя SQL-запрос INSERT. Если пользователь не существует, возвращается CommonAddResult.NotExistsUser; если операция выполнена успешно, возвращается CommonAddResult.Success; в противном случае возвращается CommonAddResult.Unknown.

Метод GetTable извлекает таблицу из базы данных, выполняя запрос SQL, и возвращает ее как объект DataTable.

Метод UpdateUserTable обновляет пользовательскую таблицу в базе данных, выполняя SQL-запрос UPDATE. Если операция выполнена успешно, возвращается CommonAddResult.Success; в противном случае возвращается CommonAddResult.Unknown.

20) Интерфейс IdbAsyncService определяет набор методов для работы с базой данных в асинхронном режиме.

Методы:

Task<bool> ExistsUserAsync(string login)

Проверяет, существует ли пользователь с указанным логином в базе данных. Возвращает true, если пользователь существует, и false в противном случае.

Task<bool> IsBannedUserAsync(string login)

Проверяет, заблокирован ли пользователь с указанным логином. Возвращает true, если пользователь заблокирован, и false в противном случае.

Task<string> GetHashPasswordAsync(string login)

Возвращает хеш пароля для пользователя с указанным логином.

Task<AddUserResult> AddUserAsync(User user)

Добавляет нового пользователя в базу данных. Возвращает результат операции (AddUserResult).

Task<DataRow?> GetDataUserByLoginAsync(string login)

Возвращает информацию о пользователе с указанным логином.

Task<CommonAddResult> AddLog(string login, LoggingType loggingType)

Добавляет запись о действии пользователя в журнал логов. Возвращает результат операции (CommonAddResult).

Task<DataTable?> GetTable(string nameTable)

Возвращает таблицу с указанным именем из базы данных.

Task<UserRole> GetUserRoleAsync(string login)

Возвращает роль пользователя с указанным логином.

Task<bool> IsBusySlotAsync(uint idSlot)

Проверяет, занят ли указанный слот. Возвращает true, если слот занят, и false в противном случае.

Task<AddSlotResult> AddSlotAsync(string employeeLogin, DateOnly slotDate, TimeOnly slotTime)

Добавляет новый слот в базу данных. Возвращает результат операции (AddSlotResult).

Task<BlockSlotResult> BlockSlotAsync(uint idSlot, string userLogin)

Блокирует указанный слот для указанного пользователя. Возвращает результат операции (BlockSlotResult).

Task<CommonAddResult> UpdateUserTableAsync(string OldLogin, DbUser NewUser)

Обновляет данные о пользователе с указанным логином. Возвращает результат операции (CommonAddResult).

Task<DataTable?> GetSlotsTableByLogin(string login)

Возвращает таблицу слотов, зарезервированных указанным пользователем.

21) Конструктор класса AuthenticationViewModel принимает один аргумент - экземпляр класса, реализующего интерфейс IDbAsyncService. Это позволяет классу AuthenticationViewModel взаимодействовать с базой данных.

Еще один конструктор класса AuthenticationViewModel принимает экземпляр класса DbConnection. Этот конструктор используется для установления соединения с базой данных. Если соединение не может быть установлено, выбрасывается исключение типа ConnectionException.

Класс AuthenticationViewModel содержит два метода: LoginAsync и RegisterAsync.

Метод LoginAsync позволяет пользователю авторизоваться в системе. Метод принимает два параметра - логин и пароль пользователя. Если логин пользователя не найден в базе данных, выводится сообщение "The user does not exist". Если пароль пользователя неверен, выводится сообщение "The password is incorrect". Если пользователь заблокирован, выводится сообщение "User is blocked". В случае успешной авторизации выводится сообщение "User has been successfully authorized".

Метод RegisterAsync позволяет пользователю зарегистрироваться в системе. Метод принимает один параметр - экземпляр класса User. Если пользователь успешно зарегистрирован, выводится сообщение "The user is successfully registered". Если пользователь с таким логином уже существует, выводится сообщение "The user with this login already exists". Если произошла ошибка в процессе регистрации, выводится сообщение "...Error...".

22) Класс Auth является страницей приложения и отвечает за авторизацию пользователя.

Поля:

\_dbAsyncService: поле типа IDbAsyncService, которое используется для взаимодействия с базой данных.

Конструкторы:

internal Auth(IDbAsyncService dbAsyncService): конструктор класса, который инициализирует поле \_dbAsyncService переданным в параметрах объектом типа IDbAsyncService.

Методы:

private void SetError(string? message): метод, который устанавливает текст ошибки в элемент AuthExceptionTextBox. Принимает на вход строку с сообщением об ошибке.

private void loginTextBox\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e): метод, который обрабатывает событие нажатия левой кнопки мыши на текстовом поле ввода логина. Очищает поле ввода, если оно содержит текст "TextInputLogin". Если при переходе в поле ввода логина поле ввода пароля осталось пустым, то метод заполняет поле ввода пароля текстом "TextInputPassword".

private void passTextBox\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e): метод, который обрабатывает событие нажатия левой кнопки мыши на текстовом поле ввода пароля. Очищает поле ввода, если оно содержит текст "TextInputPassword". Если при переходе в поле ввода пароля поле ввода логина осталось пустым, то метод заполняет поле ввода логина текстом "TextInputLogin".

private async void loginClick(object sender, RoutedEventArgs e): метод, который обрабатывает событие нажатия кнопки "Войти". Получает значения полей ввода логина и пароля, затем проверяет, существует ли пользователь с таким логином в базе данных. Если пользователь не найден, то метод устанавливает сообщение об ошибке в элемент AuthExceptionTextBox. Если пользователь найден, то метод проверяет правильность введенного пароля. Если пароль введен неверно, то метод устанавливает соответствующее сообщение об ошибке и записывает лог неудачной попытки аутентификации в базу данных. Если пароль введен верно, то метод определяет роль пользователя и перенаправляет его на соответствующую страницу приложения (административная панель или профиль пользователя), а также записывает лог успешной аутентификации в базу данных.

private void regButton\_Click(object sender, RoutedEventArgs e): метод, который обрабатывает событие нажатия кнопки "Зарегистрироваться". Открывает страницу регистрации пользователя.

private void employeeLoginButton\_Click(object sender, RoutedEventArgs e) - метод, вызываемый при клике на кнопку "Login as employee". В данном случае, он просто вызывает метод loginClick для выполнения процедуры аутентификации.

private void ChangeLanguageClick(object sender, RoutedEventArgs e) - метод, вызываемый при клике на кнопку "Change language". В зависимости от текущего языка приложения (хранится в свойстве App.language), метод меняет язык на противоположный и устанавливает новый язык с помощью UICultureService.SetCulture(), который является частью вспомогательной библиотеки Xk7.Helper.

private void SetError(string? message) - метод, устанавливающий сообщение об ошибке в поле AuthExceptionTextBox. Если сообщение message равно null, устанавливается значение "Unknown error".

private void loginTextBox\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e) - метод, вызываемый при клике на текстовое поле для ввода логина. Если поле содержит стандартный текст "TextInputLogin", он удаляется и цвет текста устанавливается на черный. Также проверяется, было ли заполнено поле для пароля, и если нет, в поле ввода пароля вставляется стандартный текст "TextInputPassword" и цвет текста устанавливается на серый.

private void passTextBox\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e) - метод, вызываемый при клике на текстовое поле для ввода пароля. Если поле содержит стандартный текст "TextInputPassword", он удаляется и цвет текста устанавливается на черный. Также проверяется, было ли заполнено поле для логина, и если нет, в поле ввода логина вставляется стандартный текст "TextInputLogin" и цвет текста устанавливается на серый.

private async void loginClick(object sender, RoutedEventArgs e) - метод, выполняющий процедуру аутентификации пользователя. Считывает логин и пароль из соответствующих текстовых полей. Затем вызывает метод \_dbAsyncService.GetDataUserByLoginAsync(login) для получения информации о пользователе из базы данных. Он выполняет запрос к базе данных и возвращает объект типа DataRow, содержащий информацию о пользователе с указанным логином. Если такого пользователя не существует, вызывается метод SetError с сообщением об ошибке из словаря UICultureService с ключом "ErrorUserNotExists". В противном случае создается объект user типа DbUser, используя фабрику UserFactory, которая преобразует DataRow в соответствующий тип. Если объект user равен null, то вызывается метод SetError с сообщением об ошибке из словаря UICultureService с ключом "UnknownError". Затем проверяется, заблокирован ли пользователь. Если пользователь заблокирован, то вызывается метод SetError с сообщением об ошибке из словаря UICultureService с ключом "UserIsBlockedError". Иначе, если введенный пароль не совпадает с паролем пользователя, вызывается метод SetError с сообщением об ошибке из словаря UICultureService с ключом "IncorrectPasswordError". Иначе, если логин и пароль верны, устанавливается значение свойства LoggedInUser и закрывается форма входа в систему.

23) Класс Registration : Page для регистрации нового пользователя в приложении. Он содержит ряд событий и методов, которые позволяют пользователю заполнять необходимые поля, а также обрабатывает ошибки, связанные с некорректным вводом данных или с отсутствием связи с базой данных.

Программа содержит окно с полями для ввода логина, пароля, имени, фамилии и даты рождения. Пользователь может вводить данные в эти поля, а затем нажать на кнопку "Зарегистрироваться". При нажатии на эту кнопку программа проверяет правильность ввода данных и отправляет их на сервер для сохранения.

Если данные были введены неправильно или произошла ошибка при сохранении в базе данных, программа выводит соответствующее сообщение об ошибке. В противном случае программа выводит сообщение об успешной регистрации и записывает соответствующую информацию в лог-файл.

24) Класс UserPanel является страницей WPF-приложения, предназначенной для отображения расписания встреч пользователя, его профиля и добавления новых встреч. Класс реализует интерфейс INotifyPropertyChanged, чтобы поддерживать уведомления об изменении свойств, которые могут повлиять на отображение данных.

Свойства:

\_dbService: хранит экземпляр сервиса доступа к базе данных, используемый для получения данных о расписании встреч и других операциях с базой данных.

\_user: хранит экземпляр пользователя, от имени которого происходят действия на странице.

Методы:

CreateAsync(): создает экземпляр страницы UserPanel и заполняет таблицу расписания встреч пользователя с помощью данных из базы данных. Возвращает асинхронную задачу, завершающуюся экземпляром UserPanel.

ChangeLanguageClick(): обрабатывает нажатие кнопки смены языка интерфейса. Изменяет язык на противоположный и обновляет интерфейс.

ExitClick(): обрабатывает нажатие кнопки выхода из аккаунта. Перенаправляет пользователя на страницу авторизации.

AddSlotButton\_Click(): обрабатывает нажатие кнопки добавления новой встречи. Открывает диалоговое окно для выбора времени и даты встречи.

ProfileButton\_Click(): обрабатывает нажатие кнопки просмотра профиля пользователя. Перенаправляет пользователя на страницу профиля.

SlotsButton\_Click(): обрабатывает нажатие кнопки просмотра расписания встреч. Создает экземпляр страницы UserPanel и заполняет таблицу расписания встреч пользователя с помощью данных из базы данных.

Класс UserPanel представляет собой часть многокомпонентного приложения для управления расписанием встреч и профилем пользователя. Он является одной из страниц, доступных пользователю, и обеспечивает удобный интерфейс для просмотра расписания встреч, добавления новых встреч и управления профилем.

25) Перечисление AddUserResult используется для возврата результата попытки добавления нового пользователя в систему. Оно содержит три возможных значения:

Success: добавление пользователя прошло успешно.

UserExists: пользователь уже существует в системе.

Unknown: произошла неизвестная ошибка при добавлении пользователя.

26) Перечисления BlockSlotResult определяет результаты операции блокировки слота (заблокировать время для записи на приём).

Перечисление включает следующие значения:

Success: операция блокировки была успешно выполнена

UserNotExists: пользователь, который пытается заблокировать слот, не существует в базе данных

SlotNotExists: слот, который пытаются заблокировать, не существует в базе данных

Unknown: неизвестная ошибка при выполнении операции блокировки слота

27) Перечисление CommonAddResult содержит значения, представляющие результаты операции добавления в общем случае.

Возможные значения перечисления:

Success: операция добавления завершена успешно.

NotExistsUser: пользователь не существует, операция добавления не выполнена.

Unknown: результат операции добавления неизвестен.

28) Перечисление enum определяет типы для логирования событий аутентификации пользователей.

Значения перечисления:

SuccessRegistration: успешная регистрация пользователя.

SuccessAuthentication: успешная аутентификация пользователя.

FailedAuthenticationWrongPassword: неудачная аутентификация пользователя из-за неверного пароля.

FailedAuthenticationUserBanned: неудачная аутентификация пользователя из-за блокировки аккаунта.

29) Перечисление UserRole содержит список ролей пользователей в системе. Роли включают в себя "User" (пользователь), "Psychologist" (психолог), "CopyWriter" (копирайтер), "Admin" (администратор) и "SuperAdmin" (суперадминистратор). Каждая роль представлена целым числом, которое может использоваться в коде для идентификации роли пользователя

30) Пользовательский класс исключения ConnectionException является производным от стандартного класса Exception. Он создан для обработки исключительных ситуаций, связанных с ошибками при подключении к серверу или базе данных.

Класс определяет один конструктор, который принимает строку сообщения в качестве параметра. Этот конструктор передает сообщение своему базовому классу для обработки.

ConnectionException может быть выброшен в случае, когда приложению не удается установить соединение с базой данных или при работе с сервером.

31) Внутренний класс ExecuteException, унаследованный от класса Exception. Имеет конструктор, который принимает сообщение об ошибке в качестве параметра. Это исключение может быть выброшено во время выполнения программы в случае возникновения ошибки во время выполнения некоторой операции.

32) Пользовательский класс исключения FactoryException. Исключение генерируется, когда происходит ошибка в процессе создания объекта фабрики. Класс FactoryException наследует от класса Exception, который предоставляет базовый функционал для всех исключений, возникающих во время выполнения программы. Конструктор класса принимает сообщение об ошибке в качестве параметра и передает его в базовый класс Exception.

33) Расширение метода для типа DbCommand, который добавляет параметр со значением в команду.

Класс DbCommandExtension содержит статический метод AddParameterWithValue, который добавляет параметр с заданным именем и значением в коллекцию параметров команды.

Метод принимает объект типа DbCommand в качестве первого параметра, представляющий команду, которая будет использоваться для выполнения запроса к базе данных. Второй параметр parameterName - строка, содержащая имя параметра, который будет добавлен в команду. Третий параметр parameterValue - объект, содержащий значение параметра, который будет добавлен в команду.

Когда метод вызывается, создается новый объект параметра DbParameter и устанавливаются его свойства ParameterName и Value соответственно. Затем созданный параметр добавляется в коллекцию параметров команды.