3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1 Модуль создания пользователей в системе

Данный модуль позволяет создавать, редактировать, удалять и управлять пользователями в системе.

Наглядное представление между некоторыми классами показано на диаграмме классов (см. чертеж ГУИР.400201.080 РР.3).

Основной класс, позволяющий создавать и редактировать пользователей в системе это `AdminController`. Он содержит следующие методы:

1. Метод `Index` отображает список доступных в системе пользователей для использования. Принимает в параметрах номер страницы для отображения, а также параметры для сортировки списка.
2. Метод `Create` осуществляет вывод на страницу пустых полей для создания нового объекта пользователей. Возвращает на страницу пустую модель.
3. Метод `CreatePOST` осуществляет создание нового пользователя из заполненных на странице данных. Возвращает страницу для редактирования только что созданного пользователя.
4. Метод `Edit` реализует открытие страницы на редактирование для выбранного пользователя. Принимает параметром идентификатор пользователя. Возвращает модель сущности нужного пользователя, который затем отображается на странице.
5. Метод `EditPOST` осуществляет сохранение отредактированного пользователя в систему. Принимает параметром идентификатор пользователя. Возвращает ссылку на страницу со списком всех ранее созданных пользователей.
6. Метод `Delete` осуществляет удаление заданного пользователя из системы. Принимает параметром идентификатор пользователя для удаления. Возвращает ссылку на страницу со всеми пользователями в системе.
7. Метод `SendChallengeEmail` позволяет отправить email созданному пользователю. Принимает параметром идентификатор пользователя. Возвращает ссылку на список всех существующих пользователей в системе.
8. Метод `Approve` позволяет подтвердить создание пользователя в системе. Это вводится специально, в качестве двойной проверки на безопасность, когда подтвердить создание новых пользователей может другой администратор. Принимает идентификатор пользователя в системе. Возвращает ссылку на список всех существующих пользователей.
9. Метод `Moderate` позволяет временно отключать пользователей, меняя им статус. Принимает параметром идентификатор пользователей.

Возвращает ссылку на страницу со всеми созданными пользователями в системе.

Класс `UserPartRecord` создан для хранения модели сущности пользователя в системе. Данный класс имеет следующий набор свойств:

– свойство `UserName` позволяет хранить логин пользователя;

– свойство `Email` позволяет хранить email для пользователя для отправки ему писем по необходимости системой;

– свойство `Password` позволяет хранить пароль для пользователя в зашифрованном виде;

– свойство `HashAlgorithm` позволяет задавать алгоритм, по которому будет захеширован и сохранен пароль в базе данных;

– свойство `PasswordSalt` предоставляет хэш пароля для позможности проверки валидности при авторизации пользователя в систему;

– свойство `RegistrationStatus` позволяет хранить статус пользователя.

Класс `UserService` содержит методы, которые помогают работать с пользователями в системе:

1. Метод `VerifyUserUnicity` позволяет проверять пользователя на уникальность при создании. Принимает параметрами имя пользователя и его email. Возвращает булевское значение результата проверки.

2. Метод `SendChallengeEmail` позволяет отправлять сообщение пользователям на из email адрес. Принимает параметром объект пользователя.

3. Метод `SendLostPasswordEmail` позволяет отправлять письмо с возможностью изменить пароль, если пользователь его забыл. Принимает параметрами объект пользователя и email адрес, на который нужно отправить письмо. Результатом возвращает булевское значение статуса отправленного письма.

4. Метод `ValidateLostPassword` позволяет проверить пользователя, восстановившего свою учетную запись на валидность. Принимает параметрами закодированный пароль. Возвращает объект пользователя.

3.2 Модуль создания ролей в системе

Данный модуль позволяет создавать роли, применять их на пользователей, а также осуществлять доступ к остальным модулям системе согласно заданным параметрам доступа для конкретного пользователя.

Основной класс `AdminController` осуществляет данное управление и содержит следующие методы:

1. Метод `List` выводит все роли, которые существуют в системе и были ранее созданы. Параметрами принимает страницу, которую необходимо вывести и параметры сортировки. Возвращает модель,

содержащую коллекцию ролей, которые удовлетворяют заданным параметрам.

2. Метод `Create` создает пустую модель для отображения на странице пользователя для создания новой роли. Возвращает модель, которая должна быть заполнена необходимой информацией.

3. Метод `CreatePost` создает новую сущность роли на основе введенных пользователем данных. Возвращает ссылку на страницу редактирования данной сущности или ошибку, если введенные пользователем данные некорректны.

4. Метод `Edit` осуществляет открытие страницы для редактирования существующей роли. Принимает параметром уникальный идентификатор роли для редактирования. Возвращает модель, представляющую запрашиваемую роль.

5. Метод `EditPost` осуществляет сохранение отредактированной сущности роли в системе. Принимает уникальный идентификатор роли. Возвращает ссылку на страницу редактирования этой роли или ошибку, если введенные пользователем данные некорректны.

6. Метод `Delete` позволяет удалить выбранную роль из системы. Принимает параметром уникальный идентификатор роли. Возвращает ссылку на страницу, на которой выводится список всех существующих ролей в системе.

Класс `RoleRecord` создан для хранения модели сущности роли в системе. Данный класс имеет следующий набор свойств:

– свойство `Id` хранит уникальный идентификатор роли в системе;

– свойство `Name` хранит название роли;

– коллекция `RolesPermissions` хранит в себе набор разрешений, по которым осуществляется доступ к остальным модулям системы.

Класс `PermissionRecord` создан для хранения настроек для разрешения по определенным модулям системы. Он имеет следующий набор свойств:

– свойство `Id` хранит уникальный идентификатор разрешения;

– свойство `Name` хранит имя разрешения в системе;

– свойство `Description` храни краткое описание разрешения;

– свойство `FeatureName` хранит имя модуля, к которому применяется данное разрешение.

Класс `RoleService` предоставляет удобный интерфейс для взаимодействия системы с ролями и разрешениями. Данный класс содержит следующие методы:

1. Метод `GetRoles` позволяет получить все роли в системе. Возвращает коллекцию объектов ролей.

2. Метод `GetRole` позволяет получить конкретно одну роль. Параметром принимает уникальный идентификатор роли. Возвращает объект роли.

3. Метод `GetRoleByName` позволяет получить из базы данных роль по ее имени. Параметром передается имя роли. Возвращается найденный объект роли.

4. Метод `CreateRole` создает роль в системе по имени. Параметром передается имя создаваемой роли.

5. Метод `CreatePermissionForRole` задает для существующей в системе роли соответствующее разрешение. Параметрами принимаются имя роли и имя разрешения.

6. Метод `UpdateRole` позволяет полностью обновить роль в системе. Параметром передается уникальный идентификатор роли, имя роли, и набор разрешений, которые необходимо к ней применить.

7. Метод `GetFeatureName` позволяет получить имя модуля, к которому применено данное разрешение. Параметром передается название разрешения. Возвращается название модуля.

8. Метод `GetPermissionDescription` позволяет получить описание разрешения по его имени. Параметром передается имя разрешения. Возвращается описание разрешения.

9. Метод `DeleteRole` позволяет удалить роль из системы. Параметром принимает идентификатор роли.

10. Метод `GetPermissionsForRole` позволяет получить все разрешения, которые применены к данной роли. Параметром принимает идентификатор роли. Возвращает коллекцию разрешений для заданной роли.

11. Метод `GetPermissionsForRoleByName` позволяет получить все разрешения, которые применены к данной роли по имени. Параметром принимает имя роли. Возвращает коллекцию разрешений для данной роли.

3.3 Модуль управления машинами в системе

Данный модуль позволяет управлять данными об импортированных в систему машинах и изменять их параметры при необходимости.

Класс `CarAdminController` позволяет отображать все импортированные в систему машины в виде списка и изменять их параметры. Имеет следующие методы:

1. Метод `CarsList` отображает все множество машин в системе в виде списка. Принимает параметром номер страницы для отображения, а также пользовательские параметры поиска. Возвращает список машин для отображения на страницы.

2. Метод `Create` позволяет при необходимости вручную создать машину в системе. Возвращает пустую модель для отображения на странице.

3. Метод `CreatePOST` осуществляет сохранение введенных при создании параметров машины. Возвращает ссылку на список всех существующих в системе машин.

4. Метод `Edit` позволяет редактировать параметры выбранной машины. В качестве аргумента принимает идентификатор машины для редактирования. Возвращает модель найденной машины для отображения на странице.

5. Метод `EditPOST` осуществляет сохранение отредактированных данных о машине. Параметром принимает идентификатор редактируемой машины. Возвращает ссылку на список всех машин в системе.

6. Метод `Delete` позволяет удалить выбранную машину из систему. Параметром принимает идентификатор машины. Возвращает ссылку на список машин.

7. Метод `AddProductImage` позволяет добавить к существующей машины ее изображение. Параметром принимает идентификатор изображения и объект загружаемого файла. Возвращает ссылку на загруженное изображение.

8. Метод `DeleteProductImages` позволяет удалять выбранные изображения для машины. Параметром принимает идентификатор изображения и ссылку на него. Возвращает ссылку на оставшиеся изображения для машины. Имеет перегруженную версию для удаления сразу нескольких изображений для машины. При этом в параметре передается массив ссылок для удаления изображений.

9. Метод `Validation` проверяет, чтобы создаваемая в системе машина имела уникальную комбинацию их двух полей, которые необходимы для дальнейшего поиска. Параметром принимает модель создаваемой машины.

Класс `CarRecord` представляет собой модель, которая используется для представления сущности машины в базе данных и в системе. Данная модель имеет большое количество свойств. Наиболее используемые их них следующие:

– свойство `Id` хранит уникальный идентификатор машины в базе данных;

– свойство `Vin` хранит уникальный идентификатор машины, который присваивается производителем;

– свойство `StockNo` содержит идентификатор машины для поиска;

– свойство `Year` содержит год производства машины;

– свойство `Make` содержит производителя машины;

– свойство `Model` хранит модель машины;

– свойство `ModelCode` содержит код модели машины;

– свойство `Transmission` хранит тип коробки передач для машины;

– свойство `Engine` содержит тип двигателя для машины;

– свойство `Fuel` хранит тип топлива;

– свойство `Doors` содержит описание дверей для машины;

– свойство `Body` описывает тип кузова;

– свойство `Color` хранит цвет машины;

– свойство `Mileage` описывает пробег машины;

– свойство `Type` тип машины, который в основном определяется как новая или бывшая в употреблении.

Класс `CarService` содержит в себе методы, которые позволяют получать машины из базы данных по определенным параметрам, а также изменять их и создавать новые. Опишем наиболее важные и используемые методы данного класса:

1. Метод `Get` позволяет получить из базы данных одну машину. Параметром принимает идентификатор машины. Возвращает модель машины по запросу. Имеет перегруженные экземпляры, которые позволяют получить объект машины по комбинации параметров из года, производителя, модели, идентификатора производителя и идентификатора для поиска.

2. Метод `GetRecordsWithoutSorting` осуществляет получение из базы всех машин по запрашиваемым параметрам. При этом никакая сортировка не производится. Принимает параметром объект с параметрами для запроса. Возвращает коллекцию машин, которые соответствуют запросу.

3. Метод `GetRecords` осуществляет получение из базы машин по запрашиваемым параметрам, а также дополнительно выполняет сортировку по одному из параметров. Принимает в качестве аргумента объект с параметрами для выборки. Возвращает отсортированную коллекцию машин по запрашиваемым параметрам.

4. Метод `GetActiveRecords` осуществляет получение из базы все машин, которые являются активными в системе на данный момента, а также соответствуют определенным параметрам. В качестве аргумента принимает объект с параметрами для запроса. Возвращает коллекцию машин.

5. Методы `Create`, `Update`, `Delete` позволяют соответственно создать, обновить и удалить запись о конкретной машине в базе данных. Параметрами принимают объект машины для осуществления соответствующего действия над ней.

6. Метод `GetModels` позволяет получить все модели машин в системе. Возвращает коллекцию моделей машин в системе.

7. Метод `GetModelsForMake` позволяет получить модели машин для конкретного производителя. Принимает параметром производителя и возвращает коллекцию моделей для него.

8. Метод `GetMakes` позволяет получить всех производителей машин в системе. Возвращает коллекцию производителей.

9. Метод `GetQueryPriceFiltered` осуществляет получение машин по заданной цене. Параметром принимает список листов, на которых

осуществляется поиск и необходимая цена. Возвращает коллекцию найденных машин.

10. Метод `GetQueryTypeFiltered` позволяет получить машины конкретного типа. Параметрами принимает список листов для поиска, а также необходимый тип машины. Возвращает коллекцию найденных машин.

3.4 Модуль управления списками машин в системе

Данный модуль позволяет управлять списками машин в системе. С его помощью можно создавать, редактировать и удалять списки машин по определенным параметрам. Также созданные в системе списки можно выводить на разных страницах для пользователей, которые могут осуществлять поиск и фильтрацию по ним.

Класс `CarsListAdminController` осуществляет непосредственно управление списками машин из администраторской части. Он содержит следующие методы:

1. Метод `Create` позволяет создавать новый список машин в системе. Возвращает пустую модель списка для заполнения на странице.
2. Метод `CreatePOST` осуществляет создание списка по введенным параметрам на странице. Возвращает ссылку на страницу со всеми списками машин в системе.
3. Метод `Edit` позволяет редактировать списки машин. Параметром принимает идентификатор списка для редактирования. Возвращает модель списка с заполненными полями.
4. Метод `EditPOST` осуществляет сохранение отредактированного списка машин в системе. Принимает идентификатор редактируемого списка, а возвращает ссылку на страницу со всеми списками в системе.
5. Метод `DeletePOST` осуществляет удаление выбранного списка машин из системы. Принимает параметром идентификатор списка для удаления. Возвращает ссылку на страницу со всеми списками машин.
6. Метод `List` отображает на странице все созданные ранее списки машин в системе. Возвращает модель с массивом списков. Имеет перегруженную версию, которая осуществляет отображение списков по заданным параметрам поиска.
7. Метод `EditCarProjection` осуществляет редактирование страницы информации для машины для конкретного списка. Принимает параметром идентификатор страницы информации для редактирования. Возвращает модель страницы информации с заполненными полями для редактирования.
8. Метод `EditCarProjectionPOST` осуществляет сохранение отредактированной страницы информации для списка. Принимает идентификатор страницы информации. Возвращает ссылку на страницу со всеми списками машин в системе.

Класс `CarListController` содержит методы, которые осуществляют вывод на страницу определенного списка машин, а также настройку фильтров для этих машин и поиска. Основные методы данного класса:

1. Метод `Item` осуществляет вывод на страницу в компактном виде описания для одной из машин запрашиваемого списка. Принимает параметром идентификатор списка, объект с параметрами для фильтрации, а также номер странице, на котором описание должно выводиться.
2. Метод `Filter` осуществляет настройку фильтров для списка на странице. Параметром принимает идентификатор списка, объект с параметрами для списка, а также номер страницы. Возвращает объект фильтра для отображения на странице.
3. Метод `FilterCarsByListSettings` осуществляет фильтрацию машин в списке по определенным параметрам, которые задаются при его создании. Параметром принимает коллекцию машин для фильтрации, а также объект списка, на котором машины будут отображаться. Возвращает отфильтрованную коллекцию машин.
4. Метод `CarForModelList` осуществляет поиск машин для отображения на конкретном списке. Принимает параметром идентификатор списка, а также номер страницы, на котором необходимо отобразить машины. Возвращает модель, в которой содержится коллекция с машинами для отображения на странице.
5. Метод `SearchData` осуществляет поиск и фильтрацию машин из списка по заданным пользователем параметрам. В качестве агрометров принимает идентификатор списка, параметры для поиска и фильтрации, а также номер страницы для отображения. Возвращает модель с коллекцией машин, которые удовлетворяют заданным параметрам поиска.

Класс `CarListPartRecord` представляет собой модель, которая используется для хранения записей о списке машин в системе, а также для манипуляции со списками, как в базе данных, так и на страницах. Данный метод содержит большое количество свойств, которые используются для настройки списка. Основные из ни следующие:

– свойство `Description` содержит краткое описание списка с машинами;

– свойство `CarsCount` содержит общее количество машин, которые будут отображены пользователю на странице для определенного списка;

– свойство `Filter` содержит фильтр, по которому машины будут фильтровать перед отображением на странице;

– свойство `Type` содержит тип машин, которые необходимо отобразить в данном списке;

– свойство `IsShowFilter` позволяет задать фильтр для списка на странице по году выпуска машины;

– свойство IsShowModelFilter позволяет задать фильтр для списка на странице по модели машины;

– свойство IsShowMakeFilter позволяет задать фильтр для списка на странице по производителю машин;

– свойство IsShowBodyStyleFilter позволяет задать фильтр для списка на странице по типу кузова машин;

– свойство IsShowTransmissionFilter позволяет задать фильтр для списка на странице по типу коробки передач для машины;

– свойство IsShowPriceFilter позволяет задать фильтр для списка на странице по цене машины;

– свойство IsShowDealerFilter позволяет задать фильтр для списка на странице по поставщику машин;

– свойство IsShowTypeFilter позволяет задать фильтр для списка на странице по типу машины;

– свойство DealerIds задает идентификаторы поставщиков машин, которые могут быть отображены в данном списке;

– свойство Models задает модели машин, которые могут быть отображены в данном списке;

– свойство Makes задает производителей машин, которые могут быть отображены в данном списке.

Класс CarsListService предоставляет методы, которые позволяют манипулировать со списками машин в системе. Основные методы данного класса следующие:

1. Метод CarRecordById позволяет получить объект списка машин по его идентификатору. Принимает параметром идентификатор списка для поиска. Возвращает объект списка по заданному параметру.

2. Метод Get позволяет получить объект списка машин по его адресу в системе. Параметром принимает адрес списка в системе. Возвращает объект списка по заданному параметру. Имеет перегруженную версию, которая позволяет получить список по его идентификатору и версии в системе.

3. Метод Get без параметров позволяет получить коллекцию всех списков в системе. Возвращает коллекцию всех созданных списков.

4. Метод GetCarListPathByType позволяет получить адрес списка в системе по типу машин, которые на нем отображаются. Принимает параметром тип машины для поиска списка. Возвращает найденный адрес списка или пустой адрес, если список не найден.

5. Метод Delete позволяет удалить список машин из системы. Параметром принимает объект списка для удаления.

6. Метод GetForCarListPart позволяет получить объект страницы деталей для машины. Параметром принимает объект списка машин. Возвращает найденный объект страницы деталей. Имеет перегруженную

версию, которая осуществляет поиск страницы деталей по идентификатору списка машин.

7. Метод `GetCarProjectionPart` позволяет получить объект страницы деталей по его идентификатору. Принимает идентификатор страницы деталей для поиска, а также версию объекта. Возвращает объект страницы деталей машины. Имеет перегруженную версию, которая осуществляет поиск объекта страницы деталей по ее адресу в системе.

8. Метод `GetPathByCarRecord` позволяет получить адрес страницы деталей для машины по непосредственно объекту машины. Параметром принимает объект машины. Возвращает адрес найденной страницы деталей или пустой адрес, если данная страница не найдена.

9. Метод `GetCarListProjection` позволяет получить объект страницы деталей машины по его адресу. Параметром принимает адрес для поиска. Возвращает найденный объект деталей машины.

10. Метод `GetCarListPart` позволяет получить объект списка машин по его идентификатору. Параметром принимает идентификатор списка для поиска. Возвращает объект списка машин.

3.5 Модуль управления деталями для машин

Данный модуль предоставляет классы для управления страницами деталей для машин. Каждая машина со всеми ее параметрами отображается на отдельной странице деталей. Для каждого списка машин можно настроить отдельную уникальную страницу деталей для отображения на ней каждой из машин списка.

Класс `CarProjectionAdminController` предоставляет методы для манипуляции над страницами деталей машин в системе. Основные методы данного класса следующие:

1. Метод `Create` позволяет создать страницу деталей для определенного списка машин. Параметром принимает идентификатор списка машин и уникальный идентификатор объекта страницы деталей для создания. Возвращает пустую модель для заполнения на странице.

2. Метод `CreatePOST` позволяет сохранить созданную страницу деталей для машин. Параметром принимает адрес, на который будет осуществлен переход после создания страницы деталей. Возвращает страницу, на которую необходимо перейти.

3. Метод `Edit` позволяет редактировать ранее созданную страницу деталей. Параметром принимает идентификатор страницы для редактирования. Возвращает модель страницы деталей с заполненными полями.

4. Метод `EditPOST` сохраняет отредактированную страницу деталей для машин. Параметром принимает идентификатор страницы деталей и

адрес, на который нужно перейти после сохранения изменений. Возвращает страницу, на которую необходимо перейти.

Класс `CarProjectionPartRecord` представляет собой модель для страницы деталей машин, которая используется для отображения сущностей из базы, а также для манипуляции на странице. Данный класс имеет следующие свойства:

– свойство `CarListPartId` хранит идентификатор объекта списка машин, для которого отображается данная страница деталей;

– свойство `FaceBookLinks` содержит в себе ссылки на страницы в FaceBook, которые необходимо отобразить на странице деталей;

– свойство `CarProjectionGuid` хранит уникальный идентификатор страницы деталей для машин;

– свойство `Disclamer` хранит текст специального сообщения, которое будет отображено на странице пользователю при просмотре.

Класс `TwitterService` содержит методы, которые позволяют осуществлять поиск по сервису Twitter по заданным параметрам. Данный класс имеет следующие методы:

1. Метод `GetTweetsFor` позволяет получить записи с сервиса Twitter для определенной страницы деталей. Параметрами принимает объект страницы деталей для поиска, время в минутах, на которое будет закэширован данный запрос и название списка, на котором отображается данная машина. Возвращает список всех сообщений с сервиса Twitter для заданной странице деталей машины.

2. Метод `Tweets` позволяет получить коллекцию сообщение с сервиса Twitter. Параметрами принимает название списка, на котором отображается машина и секретный ключ для доступа к сервису. Возвращает список всех машин по заданным параметрам.

Класс `CarYoutobeVideoService` содержит методы, которые позволяют осуществить поиск по сервису YouTube по заданным параметрам. Данный класс имеет следующие методы:

1. Метод `GetCarsFor` осуществляет поиск видео из сервиса YouTube по заданным параметрам. В качестве аргументов принимает объект страницы деталей и время, на которое будет закэширован данный запрос и в течение которого не будет совершаться дополнительных лишних запросов в базу данных.

2. Внутренний метод `GetForReview` осуществляет непосредственно запрос и поиск по сервису YouTube. Параметрами принимает объект страницы деталей. Возвращает коллекцию объектов найденных видео.

3. Внутренний метод `Videos` осуществляет запрос на сервис YouTube по заданным параметрам. В качестве аргументов принимает имя списка, на котором отображается машина, максимальное количество видео, которые необходимо получить, а также ключи доступа к сервису. Возвращает коллекцию объектом найденных видео.

3.6 Модуль управления специальными списками машин

Данный модель предоставляет классы, которые используются для создания и управления списками машин, которые создаются в период специальных акций или рекламных компаний. Отличительной особенностью таких списков является то, что на них для каждой машины необходимо отобразить дополнительную информацию об условиях акции. Кроме того, на данных специальных списках могут быть отображены абсолютно различные машины, в том числе и машины из разных простых списков.

Класс `SpecialAdminController` предоставляет методы для создания, редактирования и удаления специальных списков машин. Данный класс имеет следующие методы:

1. Метод `List` отображает все ранее созданные специальные списки машин. Параметрами принимает номер страницы для отображения, название акции, а также её статус.
2. Метод `Add` позволяет добавить новую акцию. Возвращает пустую модель для акции.
3. Внутренний метод `SetDatesForView` позволяет установить даты начала и окончания для сущности акции. Параметром принимает модель сущности акции.
4. Метод `AddPost` осуществляет сохранение созданной акции. Параметром принимает модель акции. Возвращает страницу со списком всех созданных ранее акций.
5. Метод `Edit` позволяет редактировать существующую акцию. Параметром принимает идентификатор акции для редактирования. Возвращает модель сущности акции с заполненными полями.
6. Метод `EditPost` позволяет сохранить отредактированную акцию в системе. Параметром принимает идентификатор акции. Возвращает ссылку на страницу со списком всех ранее созданных акций.
7. Метод `Delete` позволяет удалить акцию из системы. Параметром принимает идентификатор акции для удаления. Возвращает ссылку на страницу со всеми ранее созданными акциями.
8. Метод `VehiclesForSpecial` позволяет определить, какие из всех существующих в системе машин подпадают под действие акции. Параметрами принимает номер страницу для отображения, идентификатор акции и параметры списка для отображения на странице. Возвращает модель, к которой содержится коллекция всех машин для данной акции.

Класс `SpecialListController` предоставляет метод `List`, с помощью которого осуществляется отображение на странице специальных списков с машинами.

Метод данного `List` принимает параметром идентификаторы акций, которые необходимо отобразить на данном списке, адрес страницу деталей

для машин данного списка, а также номер страницы для отображения. Возвращает страницу со списком машин с примененными к ним акциями.

Класс `SpecialRecord` представляет собой модель для работы с сущностью акции. Данный класс имеет большое количество свойств. Основные и часто используемые из них следующие:

– свойство `Id` хранит уникальный идентификатор акции в базе данных;

– свойство `Title` содержит название акции;

– свойство `Types` содержит типы машин, для которых применяется данная акция;

– свойство `Years` содержит года машин, для которых применяется данная акция;

– свойство `Makes` хранит производителей машин для данной акции;

– свойство `Models` хранит модели машин для данной акции;

– свойство `PriceField` содержит название цены, для которой применяется данная акция;

– свойство `PriceFrom` содержит значение цены, начиная с которой будет действовать данная акция;

– свойство `PriceTo` содержит значение цены, до которой будет снижена основная цена в рамках действия данной акции;

– свойство `ActivateDateFrom` содержит дату, с которой начинает действовать данная акция;

– свойство `ActivateDateTo` содержит дату, до которой будет действовать данная акция;

– свойство `SpecialText` хранит текст для отображения на странице в рамках действия данной акции;

– свойство `DiscountText` хранит текст скидки;

– свойство `DiscountName` хранит название скидки;

– свойство `DiscountPrice` содержит цену, которая действует в рамках данной скидки.

Класс `SpecialListRecord` представляет собой модель, которая используется для манипулирования с сущностью специального списка машин. Данный класс имеет следующие свойства:

– свойство `SpecialRecord` содержит основную акцию, которая действует на данном списке машин;

– свойство `SpecialListSpecialRecords` содержит коллекцию объектов, которые связывают специальный список машин и все акции, которые выбраны для данного списка и все еще действуют;

– свойство `CustomFormId` хранит идентификатор формы, которая будет отображена на странице, если список машин окажется пустым.

Класс `SpecialListSpecialRecord` необходим для связывания созданных в системе специальных списков с акциями. Такая сущность

необходима, так как одна акция может быть применена к нескольким спискам, а также на одном списке может действовать сразу несколько акций.

Данный класс содержит следующие свойства:

– свойство `Id` хранит уникальный идентификатор сущности в базе данных;

– свойство `SpecialRecord` хранит объект акции для сущности;

– свойство `SpecialListRecord` хранит объект специального списка для сущности.

Класс `SpecialService` предоставляет методы для поиска, создания, редактирования и удаления акций в базе данных. Основные методы данного класса следующие:

1. Метод `Get` позволяет получить акцию по ее идентификатору. Параметром принимает идентификатор акции. Возвращает объект найденной акции.

2. Метод `GelAll` возвращает все акции из базы даных по заданным идентификаторам. Параметром принимает массив идентификаторов акций. Возвращает коллекцию найденных акций.

3. Метод `Delete` удаляет акцию из базы данных. Параметром принимает идентификатор акции для удаления. Имеет перегруженную версию, которая принимает параметром объект акции для удаления.

4. Метод `Insert` добавляет акцию в базу данных. Параметром принимает объект акции для добавления.

5. Метод `Update` обновляет запись об акции в базе данных. В качестве аргумента передается объект акции для обновления.

6. Метод `GetAll` без параметров возвращает все записи об акциях из базы данных.

7. Метод `IsTitleAlredyExist` позволяет проверить название сущности акции на уникальность. Параметрами принимает идентификатор создаваемой акции и ее название. Возвращает булевское значение результата проверки.

8. Метод `GetAllLive` позволяет получить все акции, которые действую в данный момент. Метод возвращает коллекцию объектов акций, срок действия которых еще не закончился.

9. Метод `GetSpecialsForCar` позволяет получить все акции, которые действуют на конкретную машину. Параметрами принимает объект машины, а также коллекцию все существующих акций. Возвращает коллекцию акций, которые действуют на данную машину.

10. Метод `GetLiveSpecailsForCar` позволяет получить все действующие в данный момент акции для машины. Параметром принимает объект машины. Возвращает коллекцию акций, которые срок действия которых для машины еще не истек.

Класс SpecialListSpecialService предоставляет метод, которые позволяют управлять акциями для каждого конкретного специально списка машин. Данный класс имеет один метод UpdateSpecialsForContentItem, который позволяет обновить акции для списка машин. Параметрами принимает объект списка машин и массив идентификаторов акций, которые должны на нем действовать.

3.7 Модуль создания настроек для поставщиков машин

Следующие шесть модулей для конфигурирования импорта имеют типовую структуру. У них имеется основной контроллер AdminController, а также модели, позволяющие удобно хранить данные о сущностях.

Модуль создания настроек для поставщиков машин относится к части системы, осуществляющий импорт данных. Его функции заключаются в создании сущности для каждого поставщика машин с уникальными для него параметрами.

Основной класс AdminController осуществляет отображение, создание и редактирование сущности поставщика. Методы данного класса:

1. Метод Index осуществляет вывод всех существующих в системе поставщиков машин на страницу пользователя. Параметрами принимает номер страницы, которую нужно отобразить и параметры для сортировки поставщиков. Возвращает модель, в которой содержатся все сущности поставщиков машин, удовлетворяющие заданным параметрам.
2. Метод Create осуществляет вывод на страницу пустых полей для заполнения параметров поставщика. Возвращает пустую модель поставщика для дальнейшего её заполнения.
3. Метод CreatePOST осуществляет создание поставщика машин в системе после заполнения необходимых полей на страницы. Возвращает адрес страницы, на которой отображается список поставщиков машин.
4. Метод Edit осуществляет получение сущности поставщика для редактирования и вывод ее на страницу. Получает идентификатор поставщика для редактирования. Возвращает модель запрашиваемого поставщика.
5. Метод EditPOST осуществляет сохранение настроек для поставщика после его редактирования. Получает идентификатор поставщика, а возвращает список всех ранее созданных поставщиков в системе.
6. Метод Delete осуществляет удаление поставщика машин из системы. Принимает идентификатор поставщика для удаления, а возвращает адрес страницы, на которой выводится список всех поставщиков.
7. Внутренний метод AddModelError осуществляет добавление ошибки к модели, если введенные пользователем данные некорректны. Получает строковый ключ и сообщение, которое необходимо вывести.

8. Внутренний метод `IsDealerNameUnique` осуществляет проверку на то, чтобы вновь создаваемый поставщик имел уникальное имя. Принимает модель создаваемого поставщика. Возвращает булевское значение данной проверки.

Класс `DealerPart` содержит свойства, которые формируют объект модели поставщика. Данный класс имеет следующие свойства:

– свойство `DealerId` хранит идентификатор поставщика машин в системе;

– свойство `DealerName` хранит имя поставщика в системе;

– свойство `WebSiteName` содержит адрес сайта, для которого используется данный поставщик;

– свойство `WebSiteUrl` хранит URL адрес сайта;

– свойство `WebSiteLocation` содержит сетевой адрес сайта в локальной компьютерной сети;

– свойство `SiteDbServer` хранит адрес сервера, на котором развернута база для импорта данных;

– свойство `SiteDbName` хранит название базы данных на сервере;

– свойство `SiteDbUser` содержит логин пользователя к базе данных;

– свойство `SiteDbPw` содержит пароль пользователя к базе данных.

3.8 Модуль создания шаблонов для файла импорта

Данный модуль осуществляет создание, редактирование и вывод шаблонов для файлов импорта.

Основной класс данного модуля `AdminController` имеет следующие методы:

1. Метод `Index` отображает список доступных в системе шаблонов для файлов импорта. Принимает в параметрах номер страницы для отображения, а также параметры для сортировки списка.

2. Метод `Create` осуществляет вывод на страницу пустых полей для создания нового шаблона. Возвращает на страницу пустую модель.

3. Метод `CreatePOST` осуществляет создание нового шаблона из заполненных на странице данных. Возвращает страницу для редактирования только что созданного шаблона.

4. Метод `Edit` реализует открытие страницы на редактирование для выбранного шаблона. Принимает параметром идентификатор шаблона. Возвращает модель сущности нужного шаблона, которая затем отображается на странице.

5. Метод `EditPOST` осуществляет сохранение отредактированного шаблона в систему. Принимает параметром идентификатор шаблона. Возвращает ссылку на страницу со списком всех ранее созданных шаблонов.

6. Метод `Delete` осуществляет удаление заданного шаблона для файла импорта из системы. Принимает параметром идентификатор шаблона для удаления. Возвращает ссылку на страницу со всеми шаблонами в системе.

7. Метод `CreateField` осуществляет создания нового поля для шаблона файла. Данный метод выполняется в ответ на асинхронный запрос на сервер со стороны клиента. Принимает параметром идентификатор шаблона, к которому необходимо добавить новое поле. Возвращает только, что созданное поле на клиент.

8. Метод `EditField` позволяет редактировать поля шаблона. Выполняется при асинхронном запросе на сервер. Возвращает сообщение, о том, что поле успешно отредактировано, или сообщение об ошибке.

9. Метод `DeleteField` позволяет удалить поле из шаблона. Принимает параметром уникальный идентификатор поля для удаления. Возвращает сообщение об успешном удалении поля или сообщение об ошибке.

10. Внутренний метод `GetFeedTemplateFields` осуществляет получение всех полей для необходимого шаблона. Параметром принимает идентификатор шаблона. Возвращает коллекцию полей для запрашиваемого шаблона для файла импорта.

11. Внутренний метод `IsNameUnique` реализует проверку на то, что вновь создаваемый шаблон для файла импорта имеет уникальное имя. Принимает параметром модель сущности шаблона. Возвращает булевское значение результата проверки.

Класс `FeedTemplatePartRecord` содержит свойства, которые формируют модель шаблона для файла импорта:

– свойство `Name` задает имя для шаблона;

– свойство `Description` задает краткое описание шаблона;

– свойство `FileType` задает тип файла импорта.

Класс `FeedTemplateFieldPartRecord` содержит свойства, которые формирую модель сущности поля для шаблона:

– свойство `FeedTemplateId` содержит идентификатор шаблона, которому принадлежит данное свойство;

– свойство `FieldName` хранит имя поля;

– свойство `Comment` хранит комментарий для описания данного поля.

3.9 Модуль сопоставления полей файла импорта

Данный модуль позволяет создавать сопоставление для полей, которые были созданы в шаблоне файла с существующими в базе данных полями. Также он позволяет редактировать и удалять сущности сопоставления.

Основной класс `AdminController` осуществляет данное управление и содержит следующие методы:

1. Метод `List` выводит все сопоставления, которые существуют в системе и были ранее созданы. Параметрами принимает страницу, которую необходимо вывести и параметры сортировки. Возвращает модель, содержащую коллекцию сопоставлений, которые удовлетворяют заданным параметрам.
2. Метод `Create` создает пустую модель для отображения на странице пользователя для создания нового сопоставления. Возвращает модель, которая должна быть заполнена необходимой информацией.
3. Метод `CreatePost` создает новую сущность сопоставления на основе введенных пользователем данных. Возвращает ссылку на страницу редактирования данной сущности или ошибку, если введенные пользователем данные некорректны.
4. Метод `Edit` осуществляет открытие страницы для редактирования существующего сопоставления. Принимает параметром уникальный идентификатор сопоставления для редактирования. Возвращает модель, представляющую запрашиваемое сопоставление.
5. Метод `EditPost` осуществляет сохранение отредактированной сущности сопоставления в системе. Принимает уникальный идентификатор сопоставления. Возвращает ссылку на страницу редактирования этого сопоставления или ошибку, если введенные пользователем данные некорректны.
6. Метод `Delete` позволяет удалить выбранное сопоставление из системы. Принимает параметром уникальный идентификатор сопоставления. Возвращает ссылку на страницу, на которой выводится список всех существующих сопоставлений в системе.
7. Метод `Clone` позволяет пользователю клонировать существующее сопоставление для его дальнейшего редактирования и сохранения под новым именем. Принимает параметром уникальный идентификатор сопоставления для клонирования. Возвращает ссылку на страницу редактирования только что клонированной сущности или ошибку, если такой сущности для клонирования не существует.
8. Метод `ResetDisabledFieldMappings` реализует включение ранее отключенных полей для сопоставления. Принимает параметром уникальный идентификатор поля. Возвращает ссылку на редактирование данного поля.
9. Метод `UpdateFieldMapping` реализует обновление соответствия полей в системе, после того как пользователь выберет их на странице. Принимает параметром модель, в которой содержатся поля для обновления сопоставления. Метод выполняется при асинхронном запросе на сервер. Возвращает успешный результат запроса при отсутствии ошибок при обновлении, иначе сообщение об ошибке и ее тип.

10. Метод `DisableFieldMapping` осуществляет отключение сопоставления для выбранного поля. Метод принимает параметром уникальный идентификатор поля. Выполняется при асинхронном запросе на сервер. Возвращает сообщение об успешном выполнении или же ошибку и её тип.

Класс `MappingSetPart` используется для хранения модели сущности соответствия. Данный класс имеет следующие свойства:

– свойство `FeedTemplate` хранит в себе шаблон файла импорта, который используется для создания сопоставления;

– свойство `Name` позволяет задавать имя соответствия в системе;

– свойство `Description` позволяет задать краткое описание соответствия;

– свойство `Status` хранит в себе состояние сущности соответствия.

Класс `FieldMappingService` предоставляет различные методы для управления соответствиями между полями в системе. Данный класс содержит следующие методы:

1. Метод `FindFieldMapping` позволяет найти соответствие в системе. Принимает параметром идентификатор соответствия. Возвращает найденный объект соответствия.

2. Метод `DisableFieldMapping` позволяет отключить соответствие для заданного поля. Принимает параметром объект соответствия.

3. Метод `EnableFieldMapping` создает соответствие между полями. Принимает параметром объект соответствия.

4. Метод `UpdateFieldMapping` обновляет уже существующее в системе соответствие между полями. Принимает параметром объект соответствия.

5. Метод `ChangeTemplateField` позволяет изменить шаблон файла импорта для заданного соответствия. Параметрами принимает объект соответствия и уникальный идентификатор нового шаблона, который должен быть применен к нему.

6. Внутренний метод `IsTemplateFieldMapped` позволяет проверить соответствие на то, что для него выбран один из ранее созданных шаблонов для файлов импорта. Принимает параметрами объекты шаблона и соответствия.

3.10 Модуль создания настроек для файла импорта

Данный модуль осуществляет создание, редактирование и вывод настроек для файлов импорта. Позволяет задавать местоположение файла на сервере, его имя, а также сущность сопоставления ранее созданных полей.

Основной класс `AdminController` позволяет осуществлять данные функции и имеет следующие методы:

1. Метод `Index` отображает список всех доступных ранее созданных настроек для файла импорта в системе. Параметрами принимает номер страницы для отображения и параметры для сортировки списка.

2. Метод `Create` осуществляет вывод на страницу пустых полей для ввода в них необходимых параметров для настройки фала импорта.

3. Метод `CreatePOST` осуществляет сохранение введенных пользователем настроек со страницы непосредственно в базу данных.

4. Метод `Edit` позволяет редактировать ранее созданный файл настроек для импорта. Принимает параметром идентификатор сущности настройки и возвращает модель с сохраненными ранее данными.

5. Метод `EditPOST` осуществляет сохранение обновленных настроек для файла импорта. Параметром принимает идентификатор настройки. Возвращает ссылку на список из ранее созданных настроек.

6. Метод `Delete` реализует функциональность по удалению сущности настройки для фала импорта из системы. Параметром принимает идентификатор сущности для удаления. Возвращает ссылку на список всех настроек для файла настоек в системе.

7. Внутренний метод `IsNameUnique` осуществляет проверку на то, что вновь создаваемая сущность настройки для файла импорта имеет уникальное имя. Параметром принимает объект сущности настройки и возвращает булевское значение результата проверки на уникальное имя.

Класс `InventoryFeedFilePart` содержит свойства, которые формируют модель для сущности настройки для файла импорта:

– свойство `Name` задает имя настройки;

– свойство `FileLocation` задает пусть к файлу на диске;

– свойство `FileName` задает имя файла на диске;

– свойство `TemplateMapping` задает

3.11 Модуль создания расписания для запуска импорта

Данный модуль позволяет создавать расписание для запуска сервиса по импорту данных из файла в систему. Импорт осуществляется на основе ранее созданных настроек для файла импорта.

Основной класс `AdminController` осуществляет данное управление и содержит следующие методы:

1. Метод `Index` выводит все расписания, которые были созданы для различных фалов импорта. Параметрами принимает номер страницы, которую необходимо вывести и параметры сортировки для списка. Возвращает модель, в которой содержатся все ранее созданные расписания для импорта.

2. Метод `AdHocRun` принудительно запускает импорт для файла, не дожидаясь, времени запуска. Принимает параметром идентификатор

расписания, которое нужно запустить. Возвращает текущий статус импорта для файла. Данный метод вызывается в ответ на асинхронный запрос на сервер.

3. Метод `Create` позволяет создать новое расписание. Создает пустую модель для отображения на странице.

4. Метод `CreatePost` создает новое расписание для импорта на основе введенных пользователем данных. Возвращает ссылку на страницу со списком всех созданных ранее расписаний.

5. Метод `Edit` реализует редактирование ранее созданного расписания. Принимает параметром уникальный идентификатор расписания для редактирования. Возвращает модель с данными о запрашиваемом расписании.

6. Метод `EditPost` осуществляет сохранение изменённого расписания в системе. Параметром принимает идентификатор измененного расписания. Возвращает ссылку на страницу с отображением всех ранее созданных расписаний или ошибку, если измененные данные некорректны.

7. Метод `Delete` реализует функциональность удаления расписания из системы. Принимает параметром идентификатор расписания для удаления. Возвращает ссылку на страницу со списком всех расписаний.

8. Метод `AutoUpdate` позволяет запускать импорт для файлов по ранее заданному расписанию. Параметром принимает набор уникальных идентификаторов для запуска в строковом представлении.

9. Метод `IsShowRun` проверяет состояние расписание в системе и возвращает булевское значение результата проверки на то, что импорт для данного расписания сейчас запущен.

Класс `SchedulePart` используется для хранения модели сущности расписания. Данный класс имеет следующие свойства:

– свойство `Dealer` определяет, для какого поставщика машин составлено данное расписание импорта;

– свойство `InventoryFeedFile` определяет сущность в системе, которая хранит настройки для файла импорта;

– свойство `Time` задает время запуска импорта;

– свойство `Status` определяет текущий статус импорта для определённого файла;

– свойство `LastRun` хранит время последнего запуска импорта в системе.

Перечисление `ScheduleStatus` хранит состояние расписания для импорта и имеет следующие значения:

– значение `Waiting` выставляется для расписания, когда оно запланировано, но еще не запущено;

– значение `InQueue` задается, когда одновременно запускаются несколько импортов, и второй ожидает, пока первый завершится;

– значение `Processing` выставляется непосредственно во время импорта данных в систему;

– значение `Completed` выставляется для сразу после завершения импорта данных в систему.

3.12 Модуль статистики импорта

Данный модуль позволяет хранить, выводить и просматривать статистику по импорту данных в систему, который был произведен по расписанию или вручную. Дополнительно он позволяет отфильтровать результаты по поставщику машин или промежутку времени.

Основной класс этого модуля, как у других модулей работы с импортом `AdminController`. Он содержит следующие методы:

1. Метод `Index` осуществляет вывод статистики в виде списка на страницу. С его помощью можно просматривать статистику по импорту. Список содержит информацию о поставщиках машин, файлах импорта, количестве записей, которые содержались в файле, количестве машин, которые были импортированы, времени работы импорта и статуса завершения импорта. Метод принимает параметром номер страницу, на которой нужно отобразить данные, параметры сортировки для списка, а также опции для поиска, вводимые пользователем.

2. Метод `Details` позволяет отобразить детальную информацию о какой-либо записи статистики. Принимает параметром идентификатор запрашиваемой записи.

Класс `StatisticPart` содержит свойства, которые формируют объект модели объекта статистики. Данный класс имеет следующие свойства:

– свойство `DealerId` хранит идентификатор поставщика машин в статистике;

– свойство `DealerName` хранит имя поставщика в статистике;

– свойство `DataFeedId` хранит идентификатор объекта для файла настройки импорта;

– свойство `FeedName` хранит имя объекта для файла настройки;

– свойство `ScheduleId` хранит идентификатор объекта статистики;

– свойство `StartDate` хранит дату, когда импорт для определенного файла начался;

– свойство `EndDate` хранит дату, когда импорт для определенного файла закончился;

– свойство `CountRecords` определяет количество записей в самом файле импорта;

– свойство `CountInsert` определяет реальное число записей, которые были импортированы в систему;

– свойство `CountErrors` хранит количество ошибок, которые произошли при импорте данных из файла;

Класс `StatisticService` позволяет получать необходимую информацию о статистике для вывода ее на страницу. Данный класс имеет следующие методы:

1. Метод `GetStatisticItems` позволяет получить из базы данных все записи о статистике. Параметрами принимает название поставщика машин, и интервал времени, за который произошел импорт. Данные параметры выбираются пользователем на страницы вручную.
2. Метод `GetDealersNames` позволяет получить названия всех поставщиков машин, созданных в системе. Он необходим для того, чтобы вывести их пользователю на страницу для фильтрации всего списка.

Также в рамках данного модуля существует класс `StatisticDetailsService`, который позволяет получить более подробную информацию о записи статистики. Он имеет следующие методы:

1. Метод `GetStatisticDetails` позволяет получить более подробную информацию о записи статистики. Принимает параметром уникальный идентификатор записи.
2. Метод `GetErrorsCount` позволяет получить количество ошибок, которые произошли при импорте для данной записи статистики. Параметром принимает идентификатор запрашиваемой записи.

3.13 Модуль обработки файлов перед импортом

Класс `ArchiveExtractor` осуществляет распаковку файла в том случае, если он предоставляется поставщиком машин в виде архива. Данный класс содержит следующие методы:

1. Метод `ProcessImpl` запускает процесс обработки файла перед импортом. Принимает параметром объект обработки файла. Возвращает новый отредактированный объект обработки файла.
2. Метод `IsProcessingNeccessary` осуществляет проверку на то, что предварительная обработка для файла необходима. Принимает параметром объект обработки файла. Возвращает булевское значение результата проверки на необходимость предварительной обработки.
3. Метод `IsArchive` осуществляет проверку на то, что обрабатываемый файл является архивом. Параметром принимает путь к файлу. Возвращает булевское значение результата проверки файла на архив.
4. Метод `ExtractArchive` осуществляет разархивирование обрабатываемого архива. Параметрами принимает путь к архиву и путь к папке, в которую необходимо произвести разархивирование.
5. Метод `FindExtractedFeedFile` осуществляет поиск в папке, куда произошло разархивирование файла для импорта. Параметром принимает путь к папке с разархивированными данными. Возвращает

полный путь к найденному файлу импорта или вызывает исключение `FileNotFoundException`, если файл не был найден.

6. Метод `DeleteArchive` позволяет удалить архив после его обработки и разархивации. Параметром принимает путь к архиву для удаления. Метод осуществляет удаление или вызывает исключение в случае невозможности это сделать.

7. Метод `ExtractZipFile` позволяет разархивировать файл в формате zip. Параметрами принимает название архива для разархивирования и путь к папке для сохранения фалов.

Класс `FeedItemMapper` осуществляет создания соответствий между колонками таблицы в базе данных, и колонками, которые описаны в файле импорта. Данный класс имеет следующие методы:

1. Метод `Process` осуществляет обработку файла и создает соответствия. Принимает объект импортируемого файла. Возвращает объект, в котором записан результат создания соответствия между полями.

2. Метод `MapFields` осуществляет непосредственно связывание между полем из файла импорта и колонкой в таблице. Параметром принимает объект импортируемого файла и действие, которое необходимо произвести в случае ошибки создания соответствия. Возвращает объект файла импорта с созданными соответствиями.

3. Метод `CopyNotMappedFields` осуществляет поиск в файле полей, для которых не удалость установить соответствие в базе данных, и записывает результат в файлы логов. Принимает объект импортируемого файла до создания соответствия и после, а также действие, которое должно произойти при возникновении ошибки. Возвращает объект импорта с результатом обработки.

3.14 Модуль обработки файлов импорта

Класс `FeedImporter` осуществляет процесс управления задачами для импорта. Он необходим для того, чтобы реализовать постановку задач по импорту файлов в очередь, в случае если предыдущий импорт еще не закончен. Данный класс имеет следующие методы:

1. Метод `Import` является основным методом класса и осуществляет запуск импорта для файла. Параметром принимает объект задачи для импорта и объект результат импорта, для записи в него значения после выполнения.

2. Внутренний метод `HandleItem` осуществляет непосредственно запуск импорта для конкретного файла в очереди. Принимает параметром объект файла для осуществления импорта.

3. Внутренний метод `HandleItemErrors` предназначен для обработки ошибок, которые произошли в процессе импорта файла.

Параметром принимает коллекцию ошибок, которые были сохранены при импорте.

4. Внутренний метод `HandleException` предназначен для обработки исключительных ситуаций, которые возникли в ходе работы с файлом. Параметром принимает объект исключения.

5. Внутренний метод `CreateFinishingAction` создает отдельный процесс для действий, которые необходимо выполнить после обработки файла импорта. Данный метод необходим для того, чтобы не тормозить очередь обработки файлов. Параметрами принимает объект задачи, которая обработалась и коллекцию ресурсов, которые она занимала в процессе своего выполнения.

6. Внутренний метод `FinishProcessing` осуществляет освобождение ресурсов после выполнения импорта. Параметрами принимает объект выполненной задачи и коллекцию ресурсов, которые данная задача заняла в процессе обработки.

Основной класс данного модуля `FeedParser` осуществляет непосредственно разбор фала импорта, создание объекта на основе файла и запись его в базу данных. Класс имеет следующие методы:

1. Метод `Parse` осуществляет запуск процесса импорта для файла. Параметрами принимает объекты для чтения файла и схему файла. Возвращает результат разбора файла.

2. Метод `ReadHeader` осуществляет чтение заголовка файла, в котором содержатся необходимые колонки для импорта. Параметром принимает объект для чтения файла. Возвращает результат чтения.

3. Метод `ParseHeader` осуществляет разбор заголовка фала на колонки. Параметром принимает строку заголовка и схему файла. Возвращает результат разбора.

4. Метод `ValidateHeader` необходим для проверки того, что файл в заголовки имеет все необходимые колонки, которые настроены для его соответствия в базе данных. Параметрами принимает объект разобранного заголовка и схему файла. Возвращает коллекцию колонок, которые отсутствуют в файле.

5. Метод `ParseBody` осуществляет разбор тела файла. Параметрами принимает объект файла для чтения, схему фала и объект разобранного заголовка.

6. Метод `ParseLine` считывает одну строку из файла и осуществляет ее разбор. Параметрами принимает строку для разбора и разобранный заголовок. Возвращает объект разобранной строки.

7. Метод `CreateItem` осуществляет создание объекта на основе разобранных данных из тела файла. Параметрами принимает разобранную строку и схему файла.

Для того чтобы понять как описанные выше модули взаимодействуют друг с другом можно обратиться к диаграмме последовательности (см. чертеж ГУИР.400201.080 РР.2)

3.15 Описание связи между модулями на уровне базы данных

На модели данных представлена связь между модулями системы на уровне базы данных (см. чертеж ГУИР.400201.080 РР.1).

Данная схема данных имеет таблицы, необходимые для работы с пользователями в системе, а также таблицы, необходимые для управления машинами, списками и настройками для них.

Часть, необходимая для работы с пользователями содержит следующие таблицы:

1. Таблица `USER_PART_RECORD` необходима для хранения пользователей в базе данных.
2. Таблица `ROLE_RECORD` содержит все роли, созданные в системе. Данные роли могут быть применены к пользователям.
3. Таблица `USER_ROLES_PART_RECORD` необходима для осуществления связи многие-ко-многим для таблиц пользователей и ролей.
4. Таблица `PERMISSION_RECORD` хранит все разрешения на доступ к модулям системы, которые могут быть применены к данной роли.
5. Таблица `ROLES_PERMISSIONS_RECORD` реализует связь многие-ко-многим между таблицами ролей и разрешений.

Часть базы данных, необходимая для хранения информации о машинах, списках и специальных настройках содержит следующие таблицы:

1. Таблица `CAR_LIST_PART_RECORD` содержит записи о списках машин, которые были созданы в системе.
2. Таблица `CAR_RECORD` содержит записи обо всех машинах, импортированных в систему.
3. Таблица `VIDEO_RECORD` необходима для хранения видео, которые могут быть добавлены к каждой машине в отдельности.
4. Таблица `CAR_PROJECTION_PART_RECORD` содержит записи о страницах с деталями машин для каждого отдельного списка.
5. Таблица `SPECIAL_RECORD` содержит записи с акциями для машин.
6. Таблица `CAR_LIST_SPECIAL_RECORD` необходима для реализации связи многие-ко-многим между таблицей акций и списков машин.
7. Таблица `CAR_FIELD_NAME_RECORD` содержит названия полей, которые могут быть переопределены для машин.
8. Таблица `CAR_LIST_FIELD_NAME_RECORD` необходима для реализации связи многие-ко-многим между списками машин и полями.