**3** ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

**3.1** Модуль создания настроек для поставщиков машин

Данный модуль 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 ранее созданных 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.

**3.2** 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.

**3.3** 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.

**3.4** 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

**3.5** 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;
* 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.

**3.6** Модуль статистики импорта

Данный модуль относится к части системы, осуществляющий импорт данных. Его функции заключаются в создании и сохранении статистики для импорта.

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

1. Метод Index осуществляет вывод всех существующих в системе объектов статистики 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 статистики. Возвращает пустую модель статистики для дальнейшего её заполнения.
3. Метод CreatePOST осуществляет создание статистики в системе после заполнения необходимых полей на страницы. Возвращает адрес страницы, на которой отображается список статистики.
4. Метод Edit осуществляет получение сущности статистики для редактирования и вывод ее на страницу. Получает идентификатор статистики для редактирования. Возвращает модель запрашиваемой статистики.
5. Метод EditPOST осуществляет сохранение настроек для статистики после его редактирования. Получает идентификатор статистики, а возвращает список всех ранее созданных объектов статистики в системе.
6. Метод Delete осуществляет удаление поставщика машин из системы. Принимает идентификатор статистики для удаления, а возвращает адрес страницы, на которой выводится список всех ранее созданных объектов статистики.
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 IsStaticsticNameUnique осуществляет проверку на то, чтобы вновь создаваемая статистика имел уникальное имя. Принимает модель создаваемой статистики. Возвращает булевское значение данной проверки.

Класс StatisticPart содержит свойства, которые формируют объект модели статистики. 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 DateFrom хранит время начало импорта;
* свойство DateTo хранит время окончания импорта;
* свойство Status хранит статус, с которым закончился импорт.

**3.7** Модуль создания пользователей в системе

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

Связь между модулями на уровне базы данных показана на схеме данных (см. чертеж ГУИР.400201.080 PP.1).

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 хеш 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. Принмает параметрами закодированный пароль. Возвращает объект пользователя.

**3.8** 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.

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