Содержание

[Введение 3](#_Toc526447956)

[1 Аналитический обзор и постановка задачи 4](#_Toc526447957)

[1.1 Анализ существующих программных средств учета в пенитенциарных учреждениях 4](#_Toc526447958)

[1.2 Постановка задачи на разработку программного средства информационной системы лечебно-трудового профилактория 9](#_Toc526447959)

[2 Методы и модели, положенные в основу проекта 14](#_Toc526447960)

[2.1 Функциональная модель процесса учета лиц в лечебно-трудовом профилактории 14](#_Toc526447961)

[2.2 Выбор компонентов и технологий реализации программного средства 17](#_Toc526447962)

[3 Проектирование программного средства 18](#_Toc526447963)

[3.1 Разработка функциональной спецификации 18](#_Toc526447964)

[3.2 Разработка схемы ресурсов системы 19](#_Toc526447965)

[3.3 Разработка модели данных 20](#_Toc526447966)

[3.4 Разработка схемы базы данных 22](#_Toc526447967)

[3.5 Разработка классов 26](#_Toc526447968)

[4 Разработка и реализация программного кода 30](#_Toc526447969)

[4.1 Схема работы системы 30](#_Toc526447970)

[4.2 Алгоритмы модулей программного средства 31](#_Toc526447971)

[4.3 Методика использования программного средства 35](#_Toc526447972)

[5 Тестирование разработанного программного средства 40](#_Toc526447973)

[Заключение 43](#_Toc526447974)

[Список использованных источников 44](#_Toc526447975)

[Приложение (обязательное) Листинг программного кода 45](#_Toc526447976)

Введение

Актуальность темы дипломного проекта определяется многими факторами, среди которых следует выделить всеобщую актуализацию: во-первых, проблематики информации, информатизации основных направлений общественной и государственной жизни; во-вторых, управления во всех сферах человеческой деятельности, в том числе пенитенциарной системы, ее органах и учреждениях; в-третьих, функции исполнения наказаний, исправления и ресоциализации лиц, отбывших наказания; в-четверых, необходимо отметить недостаточную разработанность перечисленных вопросов применительно к современным условиям белорусской пенитенциарной системы.

Актуальность информационного обеспечения управления в пенитенциарных учреждениях Республики Беларусь обусловлена множественностью нерешенных задач: например, адекватного определения видов, объемов и требуемой достоверности информации; организации ее получения, создания, преобразования, предоставления пользователям; совершенствования системы информационного обеспечения пенитенциарных учреждений. В таких условиях необходимо максимально эффективно обеспечить информационные потребности сотрудников пенитенциарной системы, в том числе лечебно-трудовых профилакториев, за счет использования средств автоматизации обработки информации.

Таким образом, цель настоящего проекта: повышение эффективности процесса учета данных осужденных лиц в ЛТП.

Для достижения поставленной цели необходимо решить следующие задачи:

* провести аналитический обзор предметной области и выполнить постановку задачи;
* разработать модели предметной области;
* выполнить проектирование программного средства;
* разработать и реализовать алгоритмы программного средства;
* провести тестирование программного средства.

Объект исследования: процесс управления лечебно-трудовым профилакторием.

Предмет исследования: программное средство информационной системы лечебно-трудового профилактория.

# Аналитический обзор и постановка задачи

Анализ существующих программных средств учета в пенитенциарных учреждениях

Анализ литературных источников и источников, представленных в сети Интернет, показывает, что наиболее близкими к тематике дипломного проекта являются программные средства линейки ПТК АКУС (программно-технический комплекс автоматизированного картотечного учета спецконтингента), которые позволяют решать задачи по обеспечению текущей деятельности служб пенитенциарных учреждений в сфере документооборота за счет накопления и обработки данных об абонентах:

* ПК АКУС УИИ – предназначен для ведения специализированного учета лиц, отбывающих наказание, не связанное с лишением свободы на уровне уголовно исполнительных инспекций;
* ПТК АКУС СИЗО – предназначен для ведения специализированного учета в следственных изоляторах;
* ПТК АКУС ИК – предназначен для ведения специализированного учета осужденных, отбывающих наказание в исправительных колониях [1].

Также с учетом того, что лечебно-трудовые профилактории рассматриваются и как медицинские учреждения, в качестве аналогов можно рассматривать учетные медицинские системы, в частности, АИС «Стационар» и ГИС «Медицина».

Все программные средства линейки ПТК АКУС имеют знакомый пользователю интерфейс приложений Windows, допускающий работы с программой как с помощью мыши, так и с использованием клавиатуры. Рассмотрим интерфейс и возможности программных средств на примере ПТК АКУС ИК.

Главное окно программы (рисунок 1.1) состоит из следующих элементов интерфейса:

* заголовок окна, содержащий информацию о названии и версии программного средства;
* главное меню, предназначенное для выполнения команд пользователя, обеспечивающее доступ ко всем элементам программного средства;
* область панелей инструментов;
* кнопка «Пуск» с графическим меню, предназначенная для настройки специальных режимов работы программного средства;
* рабочий стол с папками прямого доступа, дублирующий пункты главного меню и предназначенный для ускорения выполнения команд и удобства работы пользователя;
* строка статуса [1].



Рисунок . – Главное окно ПТК АКУС ИК

Для доступа пользователя к данным в программе существует набор объектов для отображения и редактирования данных. Одни объекты предназначены для просмотра данных (например, надпись, таблица отображения), другие – для вывода и редактирования (поле ввода, поле выбора справочника). В особую группу объектов входят линейные и экранные формы – программные окна, которые могут содержать другие объекты отображения и редактирования данных.

Программное средство предоставляет пользователю следующие функции:

* редактирование данных справочников (добавление, изменение, удаление);
* редактирование данных учета (добавление, изменение, удаление);
* поиск данных с использованием запросов;
* вывод данных с использованием отчетов.

Справочники представляют собой структурированные таблицы данных, содержащие информацию по определенной теме (города, профессии и т.п.).

Программное средство позволяет редактировать следующие основные наборы учетных данных:

* данные осужденных;
* данные осуждения;
* данные профилактического учета;
* условия содержания;
* перемещения;
* свидания;
* выбытия осужденных;
* освобождения осужденных и так далее.

АИС «Стационар» (разработчик – БелИнформПроект) – программная система, предназначенная для автоматизации лечебных учреждений с постоянными местами [2].

Пример интерфейса программного средства приведен на рисунке 1.1.



Рисунок . – Пример интерфейса АИС «Стационар»

Для автоматизации работы АИС «Стационар» предоставляет возможность решения следующих задач:

* организация, поддержка и ведение информационной базы данных пациентов больницы;
* прием пациентов в стационар и учет работы приемного отделения;
* контроль движения пациентов по отделениям стационара;
* оформление выписки пациентов из стационара;
* ведение базы данных электронных карт стационарных больных;
* печать медицинской и статистической карт пациентов;
* обработка статистических данных и формирование отчетов;
* получение оперативной и архивной информации о лечении пациентов в медицинском учреждении;
* печать отчетных документов за день, необходимых отчетов по деятельности больницы за различные периоды времени.

ГИС «Медицина. Управление обслуживанием» – модуль госпитальной информационной системы «Медицина» компании «КомплИТ», предназначенной для оптимизации работы медицинского учреждения и автоматизации деятельности подразделений, связанных с обслуживанием пациентов, учетно-хозяйственной деятельностью и управлением [3].

Пример интерфейса ГИС «Медицина» приведен на рисунке 1.3.

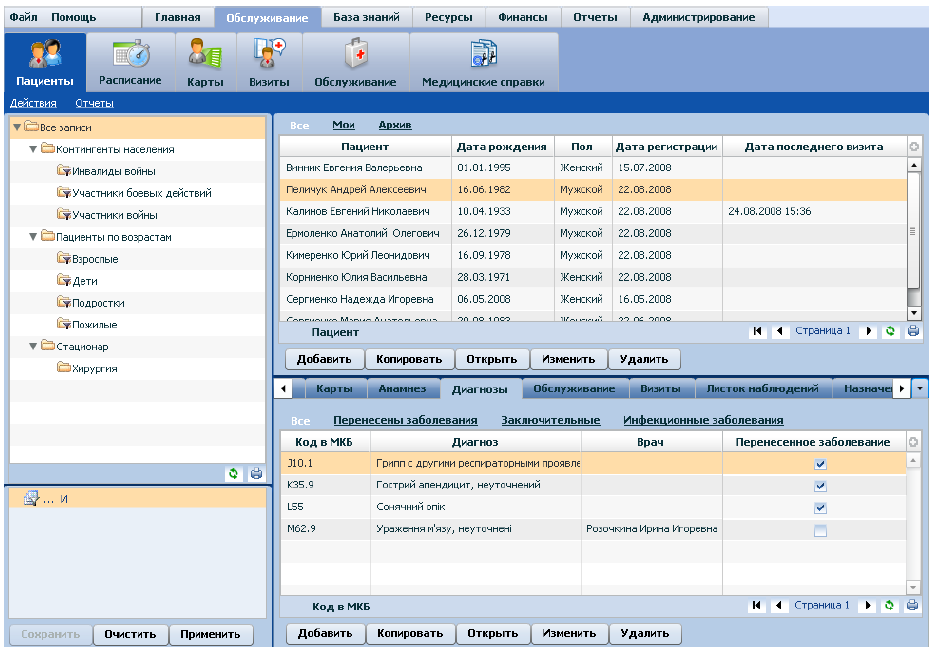


Рисунок . – Интерфейс ГИС «Медицина. Управление обслуживанием»

Для автоматизации учетной деятельности в ГИС Медицина используется модуль «Управление обслуживанием», реализующий следующие функции:

* регистрация пациентов;
* регистрация поставленных диагнозов;
* регистрация прописанных препаратов;
* регистрация предоставленных медицинских услуг;
* ведение карт пациентов;
* регистрация действий, направленных на решение вопроса, с которым обратился пациент;
* регистрация фактов проведения лечения, профилактического осмотра, диспансерного наблюдения.

Соответствие функциональных возможностей рассмотренных программных средств требованиям, предъявляемым для автоматизации ЛТП, приведено в таблице 1.1.

Таблица . – Сравнение функциональных возможностей программных средств

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | ПТК АКУС ИК | АИС Стационар | ГИС Медицина |
| Ведение базовых справочников | реализовано полностью | реализовано полностью | реализовано полностью |
| Ведение справочников, специфичных для исправительных учреждений | реализовано полностью | не реализовано | не реализовано |
| Ведение справочников, специфичных для медицинских учреждений | не реализовано | реализовано полностью | реализовано полностью |
| Ведение карточки осужденного / пациента | реализовано полностью | реализовано полностью | реализовано полностью |
| Регистрация данных, специфичных для исправительных учреждений | реализовано полностью | не реализовано | не реализовано |
| Регистрация данных, специфичных для медицинских учреждений | не реализовано | реализовано полностью | реализовано полностью |
| Поиск лиц | реализовано полностью | реализовано полностью | реализовано полностью |
| Вывод уведомлений об окончании срока | не реализовано | не реализовано | не реализовано |
| Отправка сообщения администратору системы | не реализовано | не реализовано | не реализовано |
| Разграничение прав пользователей | реализовано полностью | реализовано полностью | реализовано полностью |

Проведенный анализ существующих программных средств для учета осужденных в исправительных учреждениях и учета пациентов в медицинских учреждениях показывает, что рассмотренные программные средства не учитывают особенности учета в учреждениях типа «Лечебно-трудовой профилакторий», а именно:

* программные средства для учета осужденных в исправительных учреждениях не позволяют вести справочники и регистрировать показатели, специфичные для медицинских учреждений;
* программные средства для учета пациентов в медицинских учреждениях не позволяют вести справочники и регистрировать показатели, специфичные для исправительных учреждений.

При этом ведение базовых справочников, ведение карт лиц, их поиск, а также разграничение прав пользователей рассмотренные программные продукты реализуют.

Поскольку ЛТП является и медицинским, и исправительным учреждением одновременно, то для организации учета требуется разработка программного средства на базе современных технологий, учитывающего особенности учета лиц (осужденных, пациентов) в учреждениях типа «Лечебно-трудовой профилакторий».

Постановка задачи на разработку программного средства информационной системы лечебно-трудового профилактория

Основание для разработки

Программное средство разрабатывается по личной инициативе автора по согласованию с руководителем дипломного проекта в соответствии с приказом по университету БГУИР №112-C от 06.07.2018.

### Назначение разработки

Программное средство предназначено для учета данных лиц (осужденных, пациентов) в информационной системе лечебно-трудового профилактория.

Цель автоматизации: повышение эффективности процесса учета данных в ЛТП за счет сокращения временных затрат на ввод и обработку информации.

### Функциональные требования к программному средству

Программное средство должно обеспечивать автоматизацию следующих функциональных направлений:

* редактирование данных справочников;
* редактирование учетных данных заключенных;
* поиск заключенных по настраиваемому фильтру;
* просмотр учетных данных заключенных;
* вывод уведомлений пользователю об окончании заключения;
* отправка сообщений администратору системы.

Программное средство должно обеспечивать редактирование следующих справочников:

* справочник ЛТП;
* справочник областей;
* справочник судов;
* справочник районов;
* справочник населенных пунктов;
* справочник сотрудников;
* справочник оснований для направления в ЛТП;
* справочник медицинских комиссий;
* справочник документов, передаваемых при доставлении;
* справочник образований;
* справочник профессий;
* справочник специальностей;
* справочник видов связи;
* справочник видов задолженности;
* справочник видов отношений;
* справочник оснований сокращения срока;
* справочник оснований продления срока;
* справочник оснований досрочного освобождения;
* справочник информации о самовольном оставлении ЛТП.

Программное средство должно обеспечивать редактирования следующих основных учетных данных:

* данные осужденных лиц;
* данные адресов;
* данные осуждений;
* данные правонарушений осужденных;
* данные образования осужденных;
* данные профессий осужденных;
* данные специальностей осужденных;
* данные медицинских комиссий осужденных;
* данные превентивных надзоров.

Также программное средство должно обеспечивать редактирование следующих учетных данных о нахождении лица в ЛТП:

* доставление (оформление) лица;
* обжалование решения о направлении в ЛТП;
* отряд;
* задолженности лица;
* связи лица;
* трудовая деятельность до содержания в ЛТП;
* трудовая деятельность в ЛТП;
* зависимости;
* информация врача-нарколога;
* участие в общественной жизни ЛТП;
* отношение к лицам;
* информация психолога;
* дисциплинарные проступки;
* самовольное оставление ЛТП;
* документы, переданные при доставлении;
* информация о лице после освобождения из ЛТП.

Необходимо реализовать возможность просмотра всех учетных данных, относящихся к определенному лицу в одном информационном окне (карточка лица).

Необходимо реализовать возможность поиска лиц по следующим критериям:

* фамилия, имя, отчество;
* дата рождения;
* дата осуждения;
* дата освобождения.

Поиск по дате должен осуществляться по нахождению даты в заданном периоде. Поиск по фамилии, имени, отчеству должен осуществляться по вхождению фрагмента текста.

Вывод уведомлений пользователю об окончании заключения должен представлять собой список лиц, дата окончания заключения которых находится в 30 днях от текущей даты.

Пользователь программного средства должен иметь возможность отправить сообщение администратору системы. Администратор должен иметь возможность просматривать эти сообщения.

Программное средство должно предусматривать два уровня доступа:

* сотрудник ЛТП – может редактировать данные, относящиеся к своему ЛТП, формировать сообщения администратору;
* администратор – может редактировать любые данные, включая данные пользователей, а также просматривать сообщения.

### Требования к надежности

В процессе проектирования программного средства должны быть разработаны методики и средства, обеспечивающие:

* надежность хранения, использования и поддержания в актуальном состоянии информационного обеспечения после сбоев в электропитании устройств хранения информации;
* надежность функционирования программного обеспечения и его восстановление после сбоев в оборудовании;
* контроль целостности пользовательских данных на уровне значений атрибутов таблиц и связей между таблицами при изменении состояния пользовательских данных;
* настройку возможностей санкционированного доступа пользователей к ресурсам, включающие возможности для авторизации пользователей и управление правами их доступа.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

### Требования к составу и параметрам технических средств

К составу и параметрам технических средств предъявляются следующие требования:

* процессор с тактовой частотой, ГГц – 2, не менее;
* оперативная память объемом, ГБ – 1, не менее;
* свободное место на жестком диске, ГБ – 0,5, не менее;
* монитор с разрешением, пиксель – 1024х768, не менее;
* устройства ввода: клавиатура, мышь.

### Требования к информационной и программной совместимости

Программное средство должно удовлетворять следующим требованиям программной совместимости:

* локализованная версия операционной системы Windows 7 и выше;
* фреймворк DOT.NET 4.6.2;
* СУБД MS SQL Express 2016;
* веб-сервер IIS 10.

### Требования к программной документации

Проектная и эксплуатационная документация программного средства должна быть представлена в электронном и печатном видах и включать следующие документы:

* пояснительная записка к дипломному проекту;
* листинги программного кода;
* методика использования программного средства.

### Порядок контроля и приемки

Для контроля выполнения требования технического задания проводятся испытания. Порядок и состав испытания определяются программой и методикой испытаний.

Приемка готовой программного средства осуществляется государственной экзаменационной комиссией.

# Методы и модели, положенные в основу проекта

Функциональная модель процесса учета лиц в лечебно-трудовом профилактории

Разработаем функциональную модель процесса учета лиц в лечебно-трудовом профилактории в нотации IDEF0, согласно которой процесс разбивается на составные части – функции, для которых указывают входные и выходные параметры, а также управляющие воздействия и механизмы [4].

Контекстная диаграмма процесса учета лиц в лечебно-трудовом профилактории с использованием проектируемого программного средства приведена на рисунке 2.1.



Рисунок . – Контекстная диаграмма процесса учета лиц в ЛТП

Рассматриваемый процесс реализуется следующими механизмами:

* сотрудник;
* администратор;
* программное средство учета лиц в ЛТП.

На вход процесса подается следующая информация:

* нормативно-справочная информация (НСИ);
* информация о содержании лиц в ЛТП;
* значения фильтра;
* сообщение.

На выходе процесс формирует следующие формы:

* экранные формы журналов;
* карточка лица;
* список уведомлений;
* список сообщений.

Управляющее воздействие на процесс оказывают правила учета в ЛТП.

Диаграмма декомпозиции процесса учета лиц в ЛТП приведена на рисунке 2.2.



Рисунок . – Диаграмма декомпозиции процесса учета лиц в ЛТП

Функция А1 «Редактировать справочники» реализуется сотрудником и администратором с использованием программного средства учета лиц в ЛТП; получает на вход нормативно-справочную информацию; на выходе формирует данные справочников, подающиеся на вход функции А2.

Функция А2 «Редактировать журналы учета» реализуется сотрудником и администратором с использованием программного средства учета лиц в ЛТП; получает на вход информацию о содержании лиц в ЛТП; на выходе формирует экранные формы журналов, подающиеся на выход процесса, и учетные данные, подающиеся на вход функций А3, А4 и А5.

Функция А3 «Найти лиц по фильтру» реализуется сотрудником и администратором с использованием программного средства учета лиц в ЛТП; получает на вход учетные данные (из функции А2) и значения фильтра; на выходе формирует список найденных лиц, подающийся на вход функции А4.

Функция А4 «Просмотреть карточку лица» реализуется сотрудником и администратором с использованием программного средства учета лиц в ЛТП; получает на вход список найденных лиц (из функции А3) и учетные данные (из функции А2); на выходе формирует карточку лица, подающуюся на выход процесса.

Функция А5 «Просмотреть уведомления» реализуется сотрудником и администратором с использованием программного средства учета лиц в ЛТП; на вход получает учетные данные (из функции А2); на выходе формирует список уведомлений, подающийся на выход процесса.

Функция А6 «Отправить сообщение администратору» реализуется сотрудником и администратором с использованием программного средства учета лиц в ЛТП; на вход получает сообщение; на выходе формирует список сообщений, подающийся на выход процесса.

Построенная модель позволяет выделить основные функции разрабатываемого программного средства с точки зрения пользователей:

* редактировать справочники;
* редактировать журналы учета;
* найти лиц по фильтру;
* просмотреть карточку лица;
* просмотреть уведомления;
* отправить сообщение администратору.

## Выбор компонентов и технологий реализации программного средства

Программное средство должно быть выполнено в виде веб-приложения:

* СУБД MS SQL Server для размещения базы данных;
* сервер приложений на базе платформы ASP.NET для размещения клиентской части веб-приложения.

На момент разработки актуальной версией платформы DOT.NET является версия 4.6.2, актуальной версией СУБД MS SQL Server – 2016.

Для реализации клиентских приложений на платформе ASP.NET используются как правило две технологии:

* веб-формы – классический вариант веб-приложений ASP.NET, позволяющий создавать динамические веб-сайты, используя модель элементов веб-форм и управления событиями;
* ASP.NET MVC, позволяющий использовать современные подходы архитектуры Model-View-Controller [5].

Для настоящего проекта выбрана технология ASP.NET MVC по причине большей гибкости в реализации клиентской составляющей.

Взаимодействие веб-приложения и СУБД предполагается организовать с использованием технологии EntityFramework, реализующей ORM-модель доступа к данных [6].

При формировании интерфейса пользователя, выводимого в веб-браузер в виде веб-страниц, предполагается использование следующих фремфорков и библиотек:

* bootstrap 3 для адаптивной верстки веб-интерфейса [7];
* jquery – вспомогательная библиотека javascript [8].

Для разработки выбрана среда Microsoft Visual Studio 2017 Community – официальная среда разработки для платформы DOT.NET от производителя [6].

Выбранная среда разработки позволяет разрабатывать проекты с использованием выбранных технологий:

* с использованием менеджера пакетов NuGet имеется возможность встраивания в проект актуальных версий дополнительных библиотек и фреймворков, в том числе: EntityFramework, jquery, bootstrap;
* имеется возможность разработки проектов ASP.NET MVC [9].

# Проектирование программного средства

Разработка функциональной спецификации

Диаграмма вариантов использования программного средства приведена на рисунке 3.1.



Рисунок . – Диаграмма вариантов использования программного средства

Приведенная диаграмма вариантов использования описывает взаимоотношения и зависимости между группами действий в системе и действующими лицами (актерами), участвующими в процессе [10, 11, 12].

На данной схеме «Администратор» и «Сотрудник» связаны отношением обобщения, показывающим, что администратору доступны все варианты использования сотрудника.

Перечень вариантов использования программного средства и их соответствие функциям, выделенным на этапе функционального моделирования, приведен в таблице 3.1.

Таблица . – Перечень вариантов использования программного средства

|  |  |  |
| --- | --- | --- |
| Функция | Вариант использования | Актер |
| Редактировать справочники | Выбрать справочник | Сотрудник |
| Добавить запись справочника | Сотрудник |
| Изменить запись справочника | Сотрудник |
| Удалить запись справочника | Сотрудник |
| Редактировать журналы учета | Выбрать журнал учета | Сотрудник |
| Добавить запись журнала | Сотрудник |
| Изменить запись журнала | Сотрудник |
| Удалить запись журнала | Сотрудник |
| Найти лиц по фильтру | Найти лиц по фильтру | Сотрудник |
| Просмотреть карточку лица | Открыть карточку лица | Сотрудник |
| Просмотреть уведомления | Открыть список уведомлений | Сотрудник |
| Отправить сообщение администратору | Создать сообщение | Сотрудник |
| Открыть список сообщений | Администратор |
| Вспомогательные варианты использования | Редактировать данные пользователей | Администратор |
| Авторизоваться | Сотрудник, Администратор |

Перечисленные варианты использования составляют функциональную спецификацию требований к программному средству.

## Разработка схемы ресурсов системы

Схема ресурсов системы, отображающая конфигурацию блоков программного средства учета лиц в ЛТП, приведена на соответствующем чертеже (РТДП 6032.120.01 Схема ресурсов системы).

Приведенная схема разработана с учетом выбранных технологий реализации и соответствует клиент-серверной архитектуре веб-приложения:

* сервер баз данных, реализующий хранение данных;
* веб-приложение, предоставляющее интерфейс пользователю.

Сервер баз данных представлен следующими ресурсами:

* СУБД MS SQL Server 2016 – блок, обобщенно представляющий ресурсы СУБД, выбранной для хранения данных;
* база данных AISLTP.mdf;
* log-файл базы данных AISLTP\_log.mdf.

Веб-приложение представлено следующими ресурсами:

* AISLTP.dll – основная dll-библиотека веб-приложения;
* набор вспомогательных dll-библиотек веб-приложения, включая EntityFramework, Web.Helpters, Web.Mvc, Web.Razor и другие;
* Web.config – файл настроек приложения;
* Views – файлы cshtml-представлений;
* bootstrap.js, jquery.gqGrid.js, jquery.js – файлы вспомогательных библиотек javascript;
* bootstrap.css, ui.jqgrid.css – файлы каскадных таблиц стилей вспомогательных библиотек;
* site.css – стили, разработанные для приложения;
* modal.message.js – файл кода javascript, с помощью которого формируются AJAX-запросы при добавлении сообщения.

## Разработка модели данных

Для взаимодействия с базой данных предполагается использовать технологию Entity Framework и стратегию Code First, позволяющая сгенерировать базу данных на основании модели классов. Фрагмент схемы модели данных программного средства в виде диаграммы классов предметной области приведен на рисунке 3.2.

Приведенная схема включает часть классов предметной области, соответствующих тем объектам данных, учет которых необходимо реализовать в программном средстве.

Связи, приведенные на диаграмме, отражают отношение зависимости.

Класс модели данных Lico предназначен для представления данных осужденных лиц, информация по которым регистрируется с использованием программного средства. С этим классом связаны классы моделей данных справочников и журналов.



Рисунок . – Фрагмент модели данных программного средства

Также на диаграмме представлены следующие классы предметной области:

* Address – данные адреса;
* Debt – данные задолженностей;
* Discip – данные дисциплины;
* Doc – данные сопроводительных документов;
* Educ – данные об образовании;
* Lechen – данные о лечении;
* Naprav – данные направления лиц в ЛТП;
* Osv – данные освобождений лиц из ЛТП;
* Prof – данные о профессиях лиц;
* Obchest – данные об общественно-полезной деятельности;
* Objal – данные обжалований;
* Koap – данные административных нарушений;
* Medic – данные медицинских комиссий;
* LTP – данные ЛТП;
* Sotr – данные сотрудников ЛТП;
* User – данные пользователей программного средства;
* Role – данные ролей пользователей;
* Message – данные сообщений пользователей администратору.

## Разработка схемы базы данных

Описание ряда таблиц реляционной базы данных в формате MS SQL Server, сгенерированной на основании модели данных, приведено в таблицах 3.2-3.12.

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| ID | int | PK | код |
| Fam | navrchar |  | фамилия |
| Ima | navrchar |  | имя |
| Otc | navrchar |  | отчество |
| Dr | datetime |  | дата рождения |
| PolID | int | FK5 | код пола |
| Pasport | navrchar |  | паспортные данные |
| NacId | int | FK1 | код национальности |
| OblID | int | FK3 | код области |
| RnID | int | FK6 | код района |
| NpID | int | FK2 | код населенного пункта |
| Vneshnost | navrchar |  | описание внешности |
| Foto | image |  | фотография |
| PeredDoc\_ID | int | FK4 | код документа передачи |

Таблица . – Описание таблицы Licoes (Лица)

Таблица . – Описание таблицы Addresses (Адреса)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| ID | int | PK | код |
| OblID | int | FK2 | код области |
| RnID | int | FK3 | код района |
| NpID | int | FK1 | код населенного пункта |
| Ul | navrchar |  | улица |
| Dom | navrchar |  | дом |
| Korpus | navrchar |  | корпус |
| Kvartira | navrchar |  | квартира |
| Prim | navrchar |  | примечание |

Таблица . – Описание таблицы Address\_Lico (Адреса лиц)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| AddressID | int | PK, FK1 | код адреса |
| LicoID | int | PK, FK2 | код лица |

Таблица . – Описание таблицы Discips (Нарушения дисциплины)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| ID | int | PK | код |
| Dis | navrchar |  | нарушение дисциплины |
| Date | datetime |  | дата |
| Nak | navrchar |  | наказание |

Таблица . – Описание таблицы DiscipsLicoes (Нарушения дисциплины лиц)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| Discip\_ID | int | PK, FK1 | код нарушения дисциплины |
| Lico\_ID | int | PK, FK2 | код лица |

Таблица . – Описание таблицы Osvs (Освобождения)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| ID | int | PK | код |
| DateOsv | datetime |  | дата освобождения |
| DatePrib | datetime |  | дата прибытия к месту жительства |
| Control | navrchar |  | ФИО сотрудника |
| Locat | bit |  | имеется ли место жительства |
| Dost | bit |  | необходимость доставки до дому |
| Trud | navrchar |  | информация о трудоустройстве |
| Posic | navrchar |  | позиция лица о трудоустройстве |
| Bron | bit |  | бронирование рабочего места |

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| Osv\_ID | int | PK, FK2 | код освобождения |
| Lico\_ID | int | PK, FK1 | код лица |

Таблица . – Описание таблицы OsvLicoes (Освобождения лиц)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| ID | int | PK | код |
| Txt | navrchar |  | наименование |

Таблица . – Описание таблицы Nacs (Национальности)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| ID | int | PK | код |
| OsnnapId | int | FK3 | код основания направления |
| LTPID | int | FK2 | код ЛТП |
| DataPom | datetime |  | дата помещения в ЛТП |
| DataOsv | datetime |  | предполагаемая дата освобождения |
| FamD | nvarchar |  | фамилия сотрудника, доставившего лицо |
| ImaD | nvarchar |  | имя сотрудника, доставившего лицо |
| OtcD | nvarchar |  | отчество сотрудника, доставившего лицо |
| TelD | nvarchar |  | телефон сотрудника, доставившего лицо |
| FamO | nvarchar |  | фамилия сотрудника, оформившего лицо |
| ImaO | nvarchar |  | имя сотрудника, оформившего лицо |
| OtcO | nvarchar |  | отчество сотрудника, оформившего лицо |
| TelO | nvarchar |  | телефон сотрудника, оформившего лицо |
| Lico\_ID | int | FK1 | код лица |

Таблица 3.10 – Описание таблицы Napravs (Оформления лиц)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| ID | int | PK | код |
| Nom | nvarchar |  | номер (наименование) |
| Np | nvarchar |  | населенный пункт |
| Ul | nvarchar |  | улица |
| Dom | nvarchar |  | дом |
| Pindex | int |  | почтовый индекс |
| Teldej | nvarchar |  | телефон дежурного |
| Email | nvarchar |  | адрес электронной почты |

Таблица . – Описание таблицы LTPs (ЛТП)

Таблица . – Описание таблицы Messages (Сообщения)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Признак ключа | Описание |
| ID | int | PK | код |
| Date | datetime |  | дата |
| Content | nvarchar |  | содержание |
| UserId | int | FK1 | код пользователя |
| IsRead | bit |  | признак прочтения |

Фрагмент схемы базы данных приведен на рисунке 3.3.



Рисунок . – Фрагмент схемы базы данных

Схема, приведенная на рисунке 3.3, показывает таблицы реляционной базы данных, их поля и типы данных, первичные и внешние ключи, а также связи между таблицами.

## Разработка классов

Диаграмма классов показывает классы и их отношения, тем самым представляя логический аспект проекта [13, 14]. Диаграмма классов определяет типы объектов системы и различного рода статические связи, которые существуют между ними.

Диаграмма классов контроллеров программного средства, реализующих обработку данных, приведена на рисунке 3.4.



Рисунок . – Диаграмма классов контроллеров

На рисунке 3.4 приведены следующие классы:

* HomeController – контроллер стартовой страницы веб-приложения;
* AccountController – контроллер авторизации;
* UsersController – контроллер редактирования справочника пользователей;
* NotifyController – контроллер вывода уведомлений;
* SearchController – контроллер поиска лиц;
* MessagesController – контроллер управления данными сообщений администратору;
* Sp – пакет классов контроллеров редактирования справочников;
* Journal registration – пакет классов контроллеров редактирования учетных данных лиц;
* JiurnalLTP – пакет классов контроллеров редактирования учетных данных содержания лиц в ЛТП.

Пакет Sp включает следующие классы контроллеров редактирования справочников:

* CourtController – контроллер редактирования справочника судов;
* DocController – контроллер редактирования справочника видов документов;
* EducController – контроллер редактирования справочника видов образования;
* LTPController – контроллер редактирования справочника ЛТП;
* MedcomController – контроллер редактирования справочника медицинских комиссий;
* NpController – контроллер редактирования справочника населенных пунктов;
* OblController – контроллер редактирования справочника областей;
* OsndosrController – контроллер редактирования справочника оснований досрочного освобождения;
* OsnnapController – контроллер редактирования справочника оснований направления в ЛТП;
* OsnprodController – контроллер редактирования справочника оснований продления срока;
* OsnsocrController – контроллер редактирования справочника оснований сокращения срока;
* OtnoshController – контроллер редактирования справочника видов отношений;
* ProfController – контроллер редактирования справочника профессий;
* RnController – контроллер редактирования справочника районов;
* SotrController – контроллер редактирования справочника сотрудников;
* SpecController – контроллер редактирования справочника специальностей и другие.

Пакет Journal registration включает следующие классы контроллеров редактирования данных содержания лиц в ЛТП:

* AddressController – контроллер редактирования данных адресов лиц;
* EducsController – контроллер редактирования данных образования лиц;
* KoapsController – контроллер редактирования данных правонарушейни лиц;
* LicoesController – контроллер редактирования данных лиц;
* MedicsController – контроллер редактирования данных медицинской комиссий лиц;
* PriventsController – контроллер редактирования данных превентивных надзоров лиц;
* ProfsController – контроллер редактирования данных профессий лиц;
* SpecsController – контроллер редактирования данных специальностей лиц;
* UKsController – контроллер редактирования данных осуждений лиц.

Пакет JournalLTP включает следующие классы контроллеров редактирования справочников:

* DebtsController – контроллер редактирования данных задолженностей лица;
* DiscipsController – контроллер редактирования данных дисциплинарных проступков;
* DocsController – контроллер редактирования данных документов, переданных при доставлении;
* LechensController – контроллер редактирования данных лечения лиц;
* NapravsController – контроллер редактирования данных доставления (оформления) лиц;
* ObchestsController – контроллер редактирования данных участия лиц в общественной жизни;
* ObjalsController – контроллер редактирования данных обжалований решений о направлении в ЛТП;
* OsvsController – контроллер редактирования данных особождений лиц;
* OtradsController – контроллер редактирования данных отрядов;
* PsixesController – контроллер редактирования данных информации психолога;
* SamohodsController – контроллер редактирования данных самовольных оставлений ЛТП;
* SoderotnoshesController – контроллер редактирования данных отношений лиц;
* SvazsController – контроллер редактирования данных связей лиц;
* WorkLTPsController – контроллер редактирования данных трудовой деятельности в ЛТП;
* WorksController – контроллер редактирования данных трудовой деятельности до содержания в ЛТП;
* ZavisController – контроллер редактирования данных зависимостей.

Методы классов контроллеров редактирования данных рассмотрим на примере класса UsersController:

* Create() – вывод и обработка страницы добавления данных;
* Delete() – вывод страницы удаления данных;
* DeleteConfirmed() – обработка страницы удаления элемента;
* Details() – вывод страницы с подробными данными;
* Edit() – вывод и обработка страницы изменения данных;
* Index() – вывод страницы со списком данных.

Методы прочих классов контроллеров редактирования данных имеют аналогичное назначение.

Класс AccountController включает методы, формирующие страницы и обрабатывающие запросы, связанные с авторизацией пользователей:

* Index() – вывод страницы с формой авторизации;
* Login() – обработки страницы авторизации пользователя по логину и паролю;
* Logout() – обработка выхода пользователя из системы (сброс учетных данных).

Класс SearchController включает методы, формирующие страницу поиска лиц и формирование его результата:

* GetList() – формирование списка лиц на основании данных фильтра;
* Index() – вывод и обработка страницы с формой поиска.

Основываясь на разработанной проектной документации, перейдем к разработке и программной реализации алгоритмов функционирования программного средства.

# Разработка и реализация программного кода

Схема работы системы

Схема работы системы приведена на соответствующем чертеже (РТДП 6032.120.02 Схема работы системы).

Схема работы системы включает следующую последовательность операций:

1. запуск приложения и вывод главной страницы;
2. вывод формы авторизации, в которую пользователь вводит данные авторизации (логин и пароль);
3. если авторизация неудачна, то осуществляется возврат к операции авторизации;
4. если авторизация пройдена, дальнейшая работа система зависит от выбранной операции:
   1. при выборе операции «Справочники»:
      * выводятся формы редактирования выбранного справочника;
      * пользователь вводит данные выбранного справочника;
      * введенные данные сохраняются в записи справочников;
   2. при выборе операции «Журналы»:
      * выводятся формы редактирования выбранного журнала;
      * пользователь вводит данные лиц и содержания лиц в ЛТП;
      * введенные данные сохраняются в записи журналов;
   3. при выборе операции «Поиск»:
      * выводится форма поиска лиц;
      * пользователь вводит данные фильтра для отбора лиц;
      * на основании введенных данных фильтра и записей справочников и журналов формируется список лиц;
   4. при выборе операции «Сообщение»:
      * выводится форма сообщения;
      * пользователь вводит данные сообщения;
      * введенные данные сохраняются в записи сообщений;
   5. при выборе операции «Уведомления» выводится форма списка уведомлений на основании записей журналов;
   6. при выборе операции «Карточка лица» выводится форма карточки лица на основании записей справочников и журналов;
   7. при выборе операции «Выход» сеанс пользователя завершается.

## Алгоритмы модулей программного средства

Блок-схема алгоритма авторизации пользователя приведена на соответствующем чертеже (РТДП 6032.120.03 Авторизация пользователя. Схема программы). Приведенный алгоритм реализуется методами Index() и Login() класса AccountController.

Алгоритм предполагает следующую последовательность действий:

1. производится проверка того, авторизован ли пользователь:
   1. если пользователь не авторизован, то осуществляется переадресация на главную страницу;
   2. если пользователь авторизован, алгоритм продолжается;
2. выводится форма авторизации по логину и паролю;
3. из формы получаются логин и пароль, введенные пользователем;
4. осуществляется поиск пользователя из БД по логину и паролю;
5. далее производится проверка того, найден ли пользователь:
   1. если пользователь найден, то данные пользователя сохраняются в переменных сессии и происходит переадресация на главную страницу;
   2. если пользователь не найден, то формируется сообщение об ошибке авторизации и выводится форма авторизации.

Программный код, реализующий алгоритм авторизации, приведен в Приложении.

Блок-схема алгоритма формирования списка уведомлений приведена на соответствующем чертеже (РТДП 6032.120.04 Формирования списка уведомлений. Схема программы). Приведенный алгоритм соответствует методу Index() класса контроллера NotifyController.

Алгоритм предполагает следующую последовательность действий:

1. на основании данных сессии из БД извлекаются данные текущего пользователя;
2. формируется пустой список уведомлений;
3. дата начала определяется как текущая дата;
4. на основании даты начала определяется дата окончания;
5. из БД выбирается список лиц;
6. далее – в цикле для каждого лица из списка:
   1. определяется число оформлений по дате освобождения лица за период;
   2. проверка числа оформлений:
      * если больше нуля, то переход к следующему действию;
      * иначе – переход к следующему шагу цикла;
   3. получается последнее направление лица по дате освобождения за период;
   4. проверка полученного направления:
      * если направление получено, переход к следующему действию;
      * иначе – переход к следующему шагу цикла;
   5. проверка прав пользователя:
      * если пользователь имеет право на просмотр данных ЛТП выбранного лица, то переход к следующему действию;
      * иначе – переход к следующему шагу цикла;
   6. к списку уведомлений добавляется новое уведомление;
7. по окончании цикла выводится страница со списком уведомлений.

Программный код, реализующий алгоритм формирования списка уведомлений, приведен в Приложении.

Блок-схема алгоритма поиска лиц приведена на соответствующем чертеже (РТДП 6032.120.05 Поиск лиц. Схема программы). Приведенный алгоритм соответствует методам GetList() и Index() класса контроллера SearchController.

Алгоритм предполагает следующую последовательность действий:

1. выводится форма поиска;
2. извлекаются данные фильтра из формы поиска;
3. на основе записей всех лиц из БД формируется результатный список лиц;
4. если в фильтре установлен признак отбора по дате рождения:
   1. если дата окончания меньше даты начала, то дата окончания устанавливается равной дате начала;
   2. из БД извлекается список лиц с датой рождения, входящей в указанный период;
   3. результатный список переопределяется на основании полученного списка;
5. если в фильтре установлен признак отбора по дате осуждения:
   1. создается промежуточный список лиц;
   2. если дата окончания меньше даты начала, то дата окончания устанавливается равной дате начала;
   3. далее – в цикле для каждого лица из результатного списка:
      * если дата осуждения лица попадает в период, то лицо добавляется в промежуточный список;
   4. результатный список переопределяется на основании промежуточного списка;
6. если в фильтре установлен признак отбора по дате освобождения:
   1. создается промежуточный список лиц;
   2. если дата окончания меньше даты начала, то дата окончания устанавливается равной дате начала;
   3. далее – в цикле для каждого лица из результатного списка:
      * если дата освобождения лица попадает в период, то лицо добавляется в промежуточный список;
   4. результатный список переопределяется на основании промежуточного списка;
7. если в фильтре установлен признак отбора по ФИО:
   1. создается промежуточный список лиц;
   2. далее – в цикле для каждого лица из результатного списка:
      * формируется строка ФИО лица;
      * если ФИО лица содержит фрагмент ФИО из фильтра, то лицо добавляется в промежуточный список;
   3. результатный список переопределяется на основании промежуточного списка;
8. выводится результатный список.

Программный код, реализующий алгоритм поиска лиц, приведен в Приложении.

Блок-схема алгоритма формирования списка справочника приведена на соответствующем чертеже (РТДП 6032.120.06 Формирование списка справочника. Схема программы). Приведенный алгоритм соответствует методам Get() классов контроллеров справочников.

Алгоритм предполагает следующую последовательность действий:

1. из запроса получаются входные параметры;
2. если параметр сортировки не определен, то определяется сортировка «по возрастанию»;
3. на основании входных параметров определяется индекс выводимой страницы;
4. на основании входных параметров определяется число записей на странице;
5. на основании входных параметров определяется параметр отбора записей;
6. далее из БД выбирается список элементов справочника;
7. если определен параметр отбора записей, то выполняются следующие действия:
   1. создается пустой промежуточный список элементов справочника;
   2. далее – в цикле для каждого элемента справочника из списка:
      * если элемент удовлетворяет параметру отбора, то он добавляется в промежуточный список;
      * иначе – переход к следующему шагу цикла;
   3. список элементов справочника переопределяется на основе промежуточного списка;
8. определяется число записей в списке элементов справочника;
9. рассчитывается общее число страниц списка;
10. далее проверяется параметр сортировки:
    1. если сортировка «по убыванию», то список элементов справочника упорядочивается в порядке убывания элементов;
    2. иначе – упорядочивается в порядке возрастания;
11. далее формируются данные для постраничного вывода списка;
12. полученный список элементов справочника выводится пользователю.

Программный код, реализующий алгоритм формирования списка справочника, приведен в Приложении.

Блок-схема алгоритма сохранения данных лица приведена на соответствующем чертеже (РТДП 6032.120.07 Сохранение данных лица. Схема программы). Приведенный алгоритм соответствует методам Create() и Edit() класса контроллера LicoesController.

Алгоритм предполагает следующую последовательность действий:

1. из запроса получаются данные модели;
2. далее – проверяется корректность данных модели;
   1. если данные модели корректны, то выполняется следующая последовательность действий:
      * проверяется, загружен ли файл с фотографией;
      * если файл загружен, то в цикле для каждого байта файла формируется массив байтов файла с фотографией и записывается в соответствующий атрибут модели;
      * далее осуществляется поиск записи, соответствующей модели, в контексте данных;
      * если запись найдена, то она обновляется, если не найдена – добавляется к контексту данных;
      * далее сохраняются изменения контекста данных;
      * выполняется переадресация на страницу списка лиц;
   2. если данные модели не корректны, то выполняется следующая последовательность действий:
      * формируется сообщение об ошибке;
      * формируются данные списков связанных справочников;
      * выводится форма редактирования данных лиц с сообщением об ошибке.

Программный код, реализующий алгоритм сохранения данных лица, приведен в Приложении.

Методика использования программного средства

Авторизация

При переходе по адресу приложения открывается главная страница.

Для авторизации в системе необходимо выбрать пункт меню «Войти».

В результате откроется страница с формой авторизации (рисунок 4.1).

Для авторизации необходимо указать логин и пароль пользователя (в тестовой базе данных – sa и 1 соответственно) и нажать кнопку «Авторизация». В случае успешной авторизации пользователю будет доступно меню с возможность перехода к основным функциям системы (рисунок 4.2).

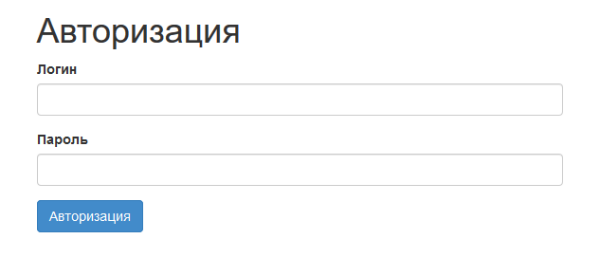


Рисунок . – Форма авторизации

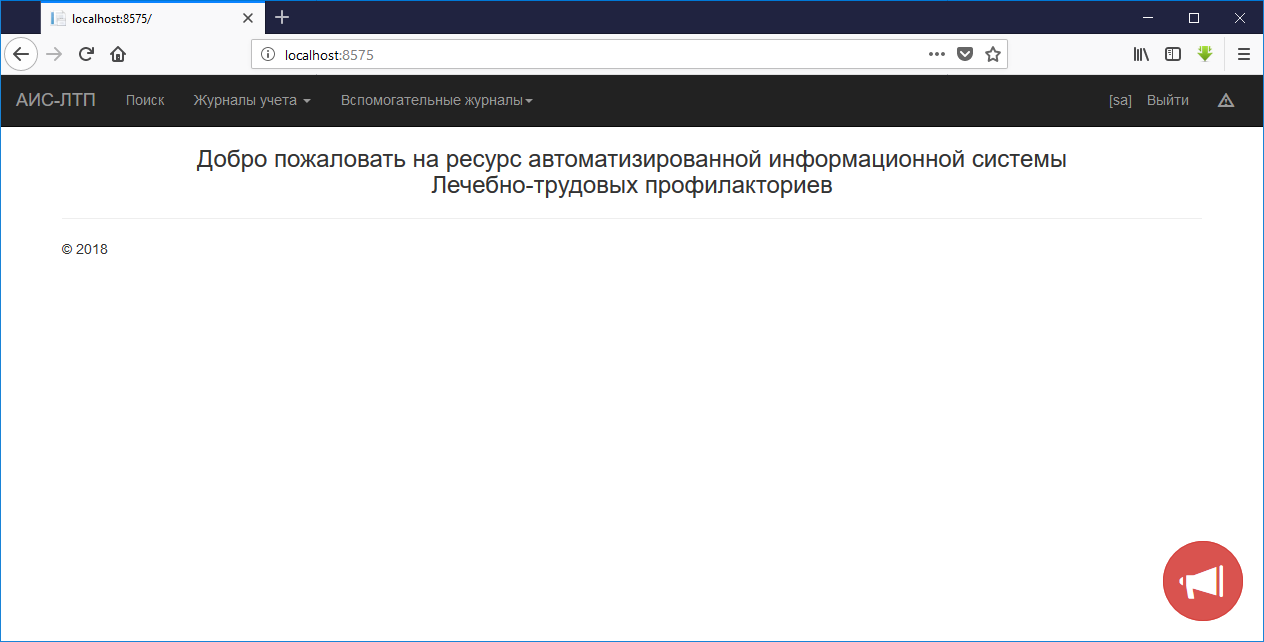


Рисунок . – Главная страница авторизованного пользователя

### Редактирование справочников

Для редактирования данных справочников необходимо в главном меню выбрать подпункт «Справочники» пункта «Вспомогательные журнал». В результате откроется список доступных справочников.

Далее необходимо выбрать справочник. В результате откроется табличная форма редактирования данных справочника (рисунок 4.3).

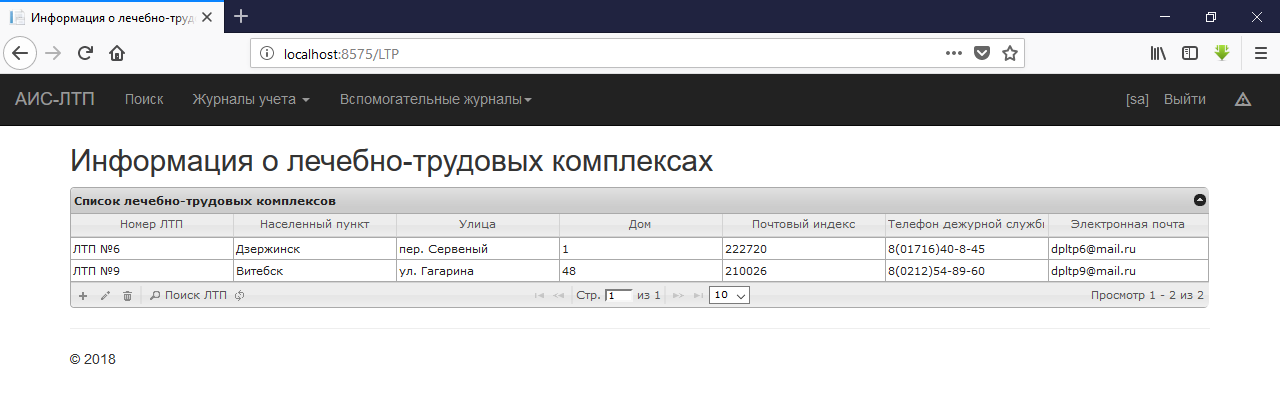


Рисунок . – Форма редактирования справочника

### Редактирование данных учета

Для редактирования данных учета необходимо выбрать необходимый подпункт пункта меню «Журналы учета». В результате откроется список выбранных учетных данных (рисунок 4.4).

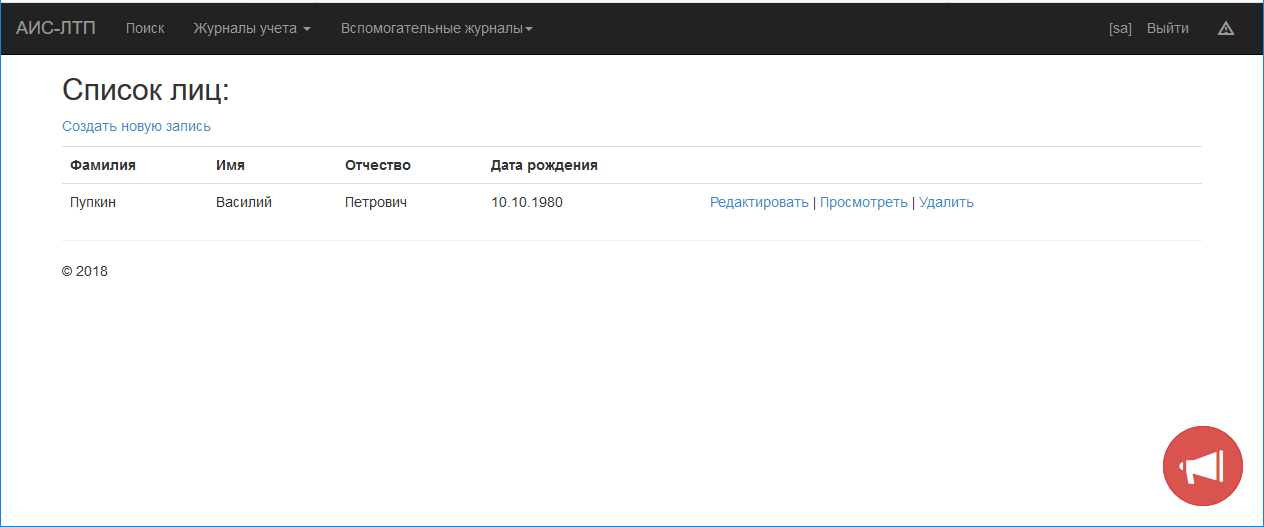


Рисунок . – Список учетных данных

Для добавления записи необходимо выбрать ссылку «Создать новую запись». Редактирование и просмотр записей осуществляется с помощью ссылок: редактировать – изменить запись; просмотреть – просмотреть детальные данные записи; удалить – удалить запись.

Форма добавления (изменения) записи приведена на рисунке 4.5.

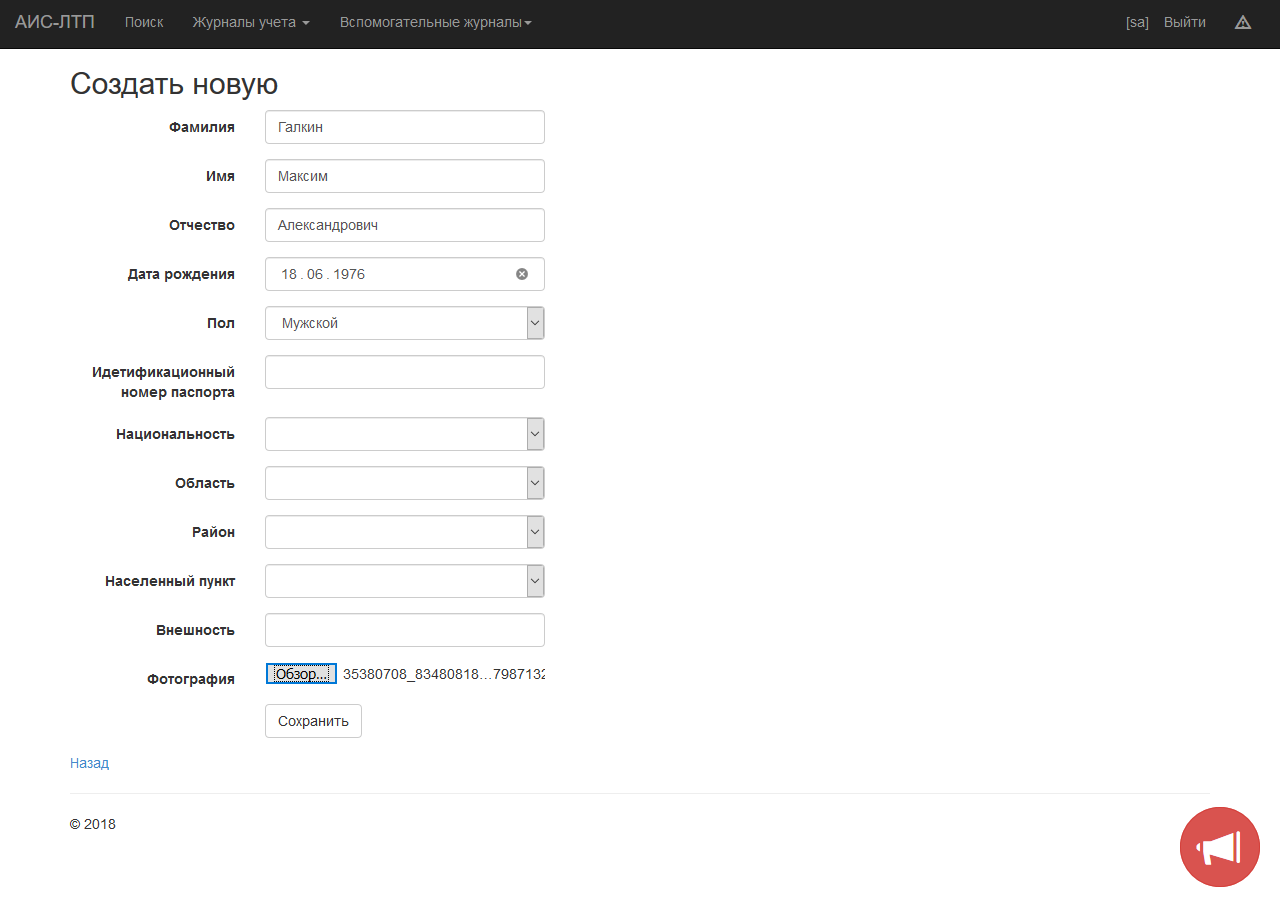


Рисунок . – Пример формы добавления (изменения) записи

Для добавления (изменения) необходимо в полях формы указать значения учетных данных и нажать кнопку «Сохранить». Для возврата к списку без сохранения необходимо нажать ссылку «Назад».

### Поиск лиц

Для поиска лиц необходимо выбрать пункт главного меню «Поиск». В результате откроется форма поиска (рисунок 4.6). В форме необходимо указать отметить необходимые условия поиска, указать данные фильтра лиц (фрагмент ФИО; период поиска по дате рождения; период поиска по дате осуждения; период поиска по дате освобождения) и нажать кнопку «Поиск». В результате будет выведен список найденных лиц.

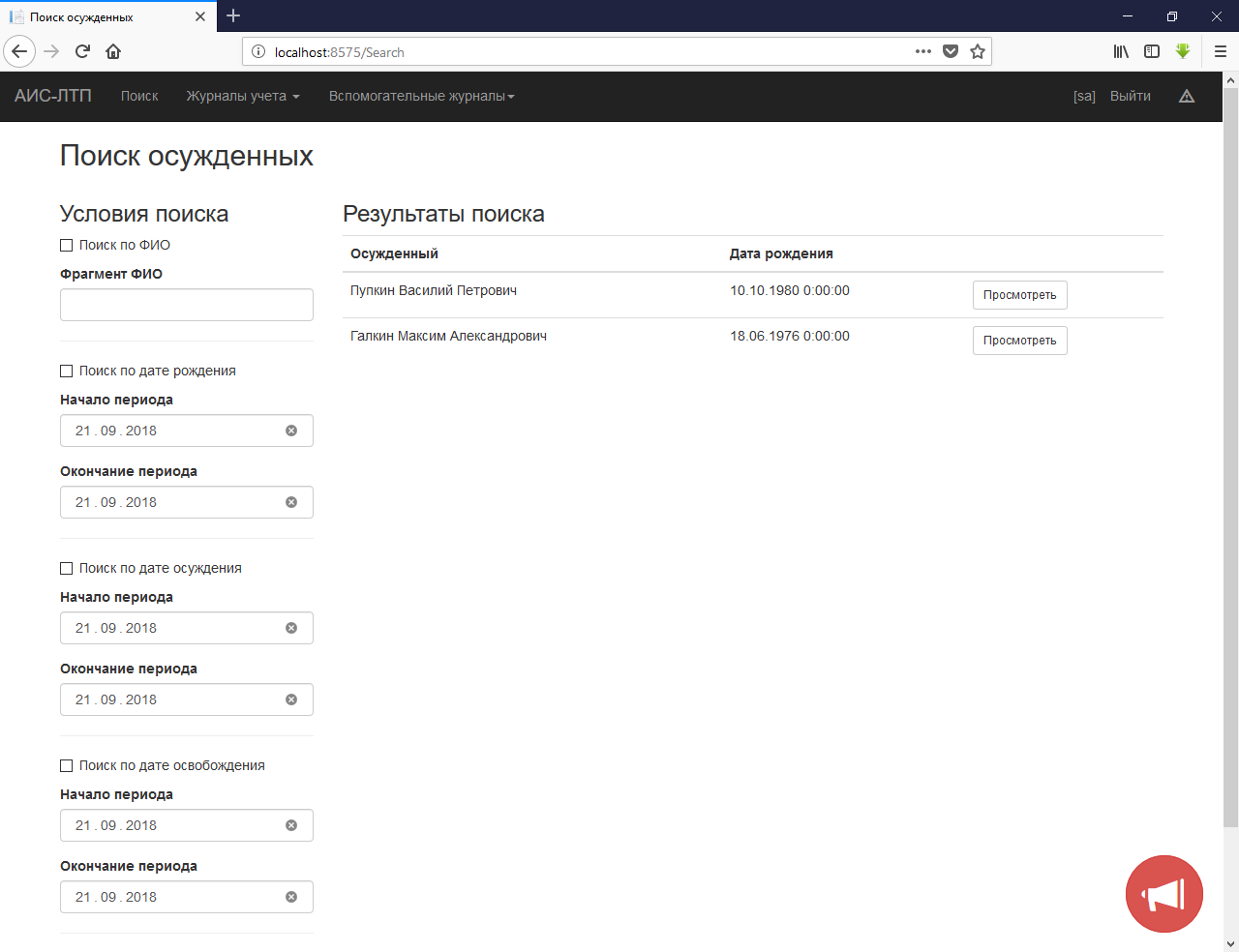


Рисунок . – Форма поиска лиц

### Отправка сообщения администратору

Для отправки сообщения администратору необходимо нажать на кнопку с изображением рупора в правом нижнем углу. В результате откроется форма создания сообщения (рисунок 4.8), в которой необходимо указать текст и нажать кнопку «Отправить».

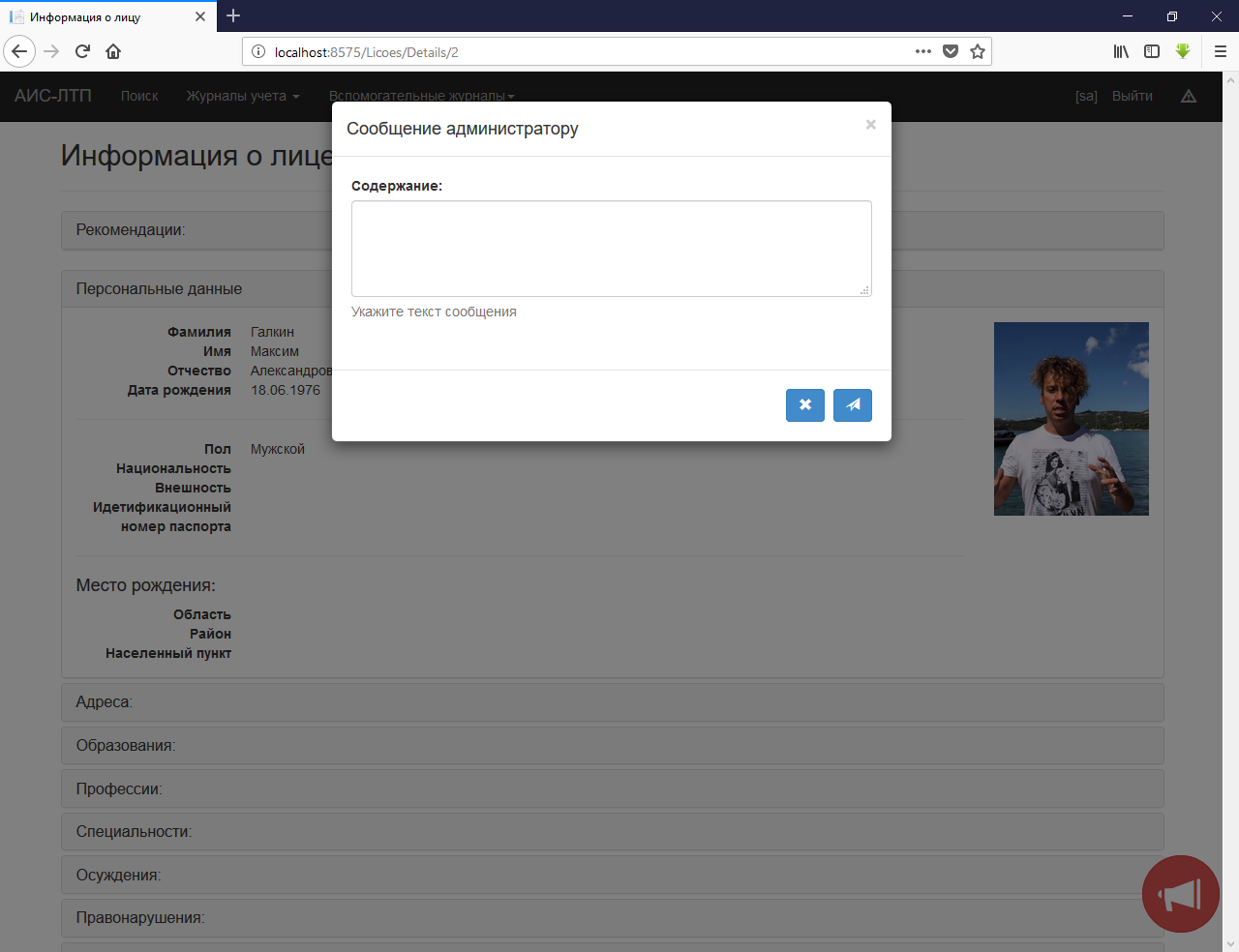


Рисунок . – Форма создания сообщения

Для просмотра сообщений администратору необходимо выбрать подпункт «Сообщения» пункта «Вспомогательные журналы» главного меню. В результате откроется список сообщений (рисунок 4.8).

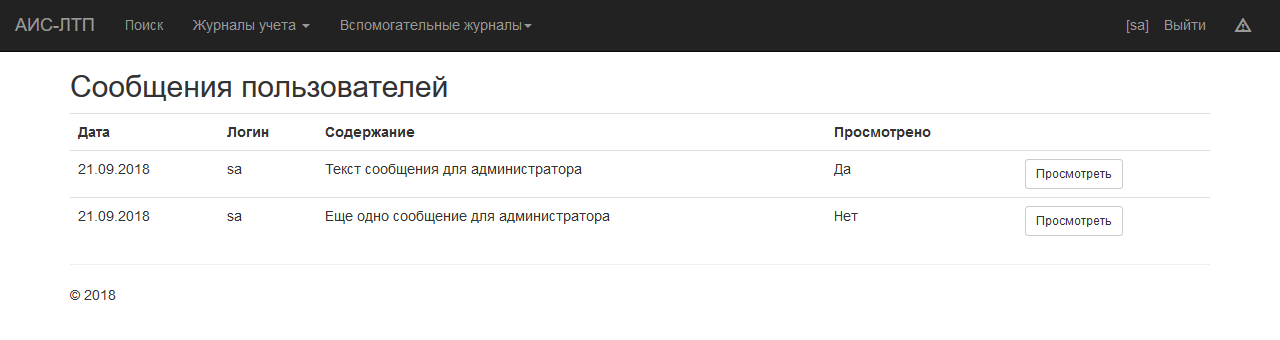


Рисунок . – Список сообщений пользователей

### Просмотр уведомлений

Для просмотра уведомлений необходимо выбрать пункт с восклицательным знаком в главном меню. В результате будет показан список актуальных уведомлений.

# Тестирование разработанного программного средства

Проверка выполнения требований к функциям программного средства проводилась на основании подготовленных тест-кейсов. При таком тестировании требуется выполнить ряд действий, при этом результат выполнения тестируемой функции программного обеспечения должен быть заранее известен. Использование заранее разработанных тест-кейсов позволяет максимально полно учесть все варианты действий пользователей и проанализировать реакцию на эти действия.

В рамках дипломного проектирования ограничимся тестами на запуск приложения и корректность выполнения основных вариантов использования.

Результаты тестирования приведены в таблице 10.

Таблица . – Результаты тестирования программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант использования (функция) / Модуль | Описание теста | Ожидаемый результат | Статус теста |
| 1 | 2 | 3 | 4 |
| Запуск приложения / Home | Набрать в адресной строке адреса веб-приложения | Вывод стартовой страницы с кнопкой «Войти» | пройден |
| Авторизоваться / Account | Запустить приложение  Нажать кнопку «Войти»  Ввести верный логин и верный пароль  Нажать кнопку «Авторизация» | Вывод стартовой страницы  Вывод формы авторизации  Вывод стартовой страницы с меню списка операций | пройден |
| Авторизоваться / Account | Запустить приложение  Нажать кнопку «Войти»  Ввести неверный логин и (или) неверный пароль  Нажать кнопку «Авторизация» | Вывод стартовой страницы  Вывод формы авторизации  Вывод формы авторизации с сообщением об ошибке авторизации | пройден |
| Открыть список уведомлений / Notify | Выбрать пункт меню «Уведомления» | Вывод страницы со списком уведомлений | пройден |
| Выбрать справочник / Контроллеры справочников | Выбрать пункт меню «Вспомогательные журналы / Справочники»  Выбрать справочник в списке | Вывод страницы со списком справочников  Вывод формы редактирования выбранного справочника | пройден |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Добавить запись справочника / Контроллеры справочников | Выбрать справочник  Нажать кнопку «Добавить новую запись»  Ввести данные записи  Нажать кнопку «Сохранить» | Вывод формы редактирования справочника  Вывод формы добавления элемента справочника  Если данные корректны, вывод сообщения об успешном добавлении, если некорректны – вывод сообщения об ошибке добавления | пройден |
| Изменить запись справочника / Контроллеры справочников | Выбрать справочник  Выбрать запись элемента в таблице  Нажать кнопку «Редактировать выбранную запись»  Ввести данные записи  Нажать кнопку «Сохранить» | Вывод формы редактирования справочника  Вывод формы изменения элемента справочника  Если данные корректны, вывод сообщения об успешном сохранении, если некорректны – вывод сообщения об ошибке сохранения | пройден |
| Удалить запись справочника / Контроллеры справочников | Выбрать справочник  Выбрать запись элемента в таблице  Нажать кнопку «Удалить выбранную запись»  Нажать кнопку «Удалить» | Вывод формы редактирования справочника  Вывод подтверждения удаления  Если на элемент нет ссылок, вывод сообщения об успешном удалении, если существуют ссылки – вывод сообщения о невозможности удаления | пройден |
| Выбрать журнал учета / Контроллеры журналов учета | Выбрать пункт меню «Журналы учета»  Выбрать журнал в списке  Выбрать лицо | Вывод страницы со списком журналов учета  Вывод списка лиц  Вывод списка данных журнала, относящихся к выбранному лицу | пройден |
| Добавить запись журнала / Контроллеры журналов учета | Выбрать журнал учета  Нажать кнопку «Добавить»  Ввести данные записи  Нажать кнопку «Сохранить» | Вывод формы списка данных  Вывод формы добавления записи журнала учета  Если данные корректны, вывод сообщения об успешном добавлении, если некорректны – вывод сообщения об ошибке добавления | пройден |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Изменить запись журнала / Контроллеры журналов учета | Выбрать журнал учета  Выбрать запись журнала  Нажать кнопку «Редактировать»  Ввести данные записи  Нажать кнопку «Сохранить» | Вывод формы списка данных  Вывод формы изменения записи журнала учета  Если данные корректны, вывод сообщения об успешном сохранении, если некорректны – вывод сообщения об ошибке сохранения | пройден |
| Удалить запись журнала / Контроллеры журналов учета | Выбрать журнал учета  Выбрать запись журнала  Нажать кнопку «Удалить»  Ввести данные записи  Нажать кнопку «Сохранить» | Вывод формы списка данных  Вывод подтверждения удаления  Если на запись нет ссылок, вывод сообщения об успешном удалении, если существуют ссылки – вывод сообщения о невозможности удаления | пройден |
| Создать сообщение / Messages | Выбрать кнопку «Создать сообщение» в правом нижнем углу страницы  Ввод данных сообщения | Вывод формы создания сообщения  Закрытие формы создания сообщения | пройден |
| Открыть список сообщений / Messages | Выбрать пункт меню «Сообщения» | Вывод страницы со списком сообщений | пройден |
| Найти лиц по фильтру / Search | Выбрать пункт меню «Поиск»  Указать данные поиска в форме фильтра  Нажать кнопку «Поиск» | Вывод формы фильтра  Вывод списка лиц согласно условиям поиска | пройден |
| Открыть карточку лица / Licoes | Найти лиц по фильтру  Нажать кноку «Просмотреть» в строке лица | Вывод страницы с подробными данными лица | пройден |
| Открыть карточку лица / Licoes | Выбрать журнал «Журнал лиц»  Нажать кноку «Просмотреть» в строке лица | Вывод страницы с подробными данными лица | пройден |

Прочие функции протестированы с использованием аналогичных тестов. Все тесты пройдены.

Заключение

В ходе дипломного проекта решены следующие задачи проектирования.

Проведен анализ существующих программных средств в пенитенциарных учреждениях, показавший наличие ряда недостатков, в том числе: рассмотренные программные средства разработаны на базе устаревших технологий – применяется платформа Visual Fox Pro, в настоящее время снятая Microsoft с поддержки; рассмотренные программные средства не учитывают особенности учета в учреждениях типа «Лечебно-трудовой профилакторий». На основании анализа разработано техническое задание на разработку программного средства учета лиц в ЛТП.

Выбраны компоненты и технологии реализации программного средства. Разработана функциональная модель процесса учета лиц в лечебно-трудовом профилактории с использованием проектируемого программного средства.

Выполнено проектирование программного средства учета лиц в ЛТП. В ходе проектирование разработаны следующие спецификации: функциональная спецификация требований в виде диаграммы вариантов использования, схема ресурсов системы, классы модели данных, схема базы данных, диаграммы классов контроллеров.

На основании проектной документации разработаны и реализованы алгоритмы функционирования программного средства. Программное средство реализовано как веб-приложение ASP.NET MVC, использующее для хранения данных СУБД MS SQL 2016. Реализация серверной части выполнена на языке программирования C#. Клиентская часть веб-приложения реализована с использованием библиотек jquery и bootstrap. Результаты тестирования показали, что разработанное программное средство удовлетворяет функциональным требованиям и функции выполняются корректно.

Дальнейшее развитие программного средства может осуществляться в следующих направлениях:

* разработка в рамках программного средства подсистемы учета назначения и выполнение медицинских процедур;
* разработка в рамках программного средства подсистемы отчетности, позволяющей формироваться различного рода отчеты как сводные, так и по отдельному ЛТП.

Список использованных источников

1. Тайгозина, Е.А. Использование ПТК АКУС в практической деятельности: методические рекомендации / Е.А. Тайгозина. – Киров: ФКУ ДПО Кировский ИПКР, 2015. – 162 с.
2. АИС «Стационар» [Электронный ресурс] / Официальный сайт. – БелИнформПроект, 2018. – Режим доступа: http://belinform.by/hospital.html.
3. ГИС Медицина. Управление обслуживанием [Электронный ресурс] / Официальный сайт. – Terrasoft, 2018. – Режим доступа: https://www.terrasoft.ru/company/news/6063.
4. Маклаков, С.В. Моделирование бизнес-процессов с AllFusion Process Modeler. – М.: Диалог-МИФИ, 2008. – 240 с.
5. Дайксрта, Т. Создание веб-проектов ASP.NET в Visual Studio [Электронный ресурс] / Т. Дайкстра. – Microsoft, 2018. – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/visual-studio/overview/2013/ web-projects.
6. Entity Framework [Электронный ресурс] / Д. Вега. – Microsoft, 2018. – Режим доступа: https://docs.microsoft.com/ru-ru/ef/ef6/.
7. Bootstrap [Электронный ресурс] / Официальный сайт. – Bootstrap, 2018. – Режим доступа: https://getbootstrap.com/docs/3.3/.
8. Jquery [Электронный ресурс] / Официальный сайт. – The jQury Foundation, 2018. – Режим доступа: https://jquery.com.
9. Visual Studio [Электронный ресурс] / Официальный сайт. – Microsoft, 2018. – Режим доступа: https://visualstudio.microsoft.com/ru/vs/.
10. Мацяшек, Л.А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML / Л.А. Мацяшек. – М.: Издательский дом «Вильямс», 2013. – 432 с.
11. Иванова, Г.С. Технология программирования: Учебник для вузов / Г.С. Иванова. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2013. – 320 с.
12. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Дж. Рамбо, А. Джекобсон. – СПб.: Питер, 2012. – 494 с.
13. Вендров, А.М. Проектирование программного обеспечения экономических информационных систем / А.М. Вендров. – М.: Финансы и статистика, 2013. – 544 с.
14. Хаф, Л. Проектирование информационных систем / Л. Хаф. – М.: Бином, 2016. – 728 с.

Приложение  
(обязательное)  
Листинг программного кода

AccountController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Security;

namespace AISLTP.Controllers

{

public class AccountController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: Account

public ActionResult Index()

{

if (HttpContext.User.Identity.IsAuthenticated)

{

return RedirectToAction("Index", "Home");

}

return View();

}

[HttpPost, ActionName("Index")]

[ValidateAntiForgeryToken]

public ActionResult Login(Auth auth)

{

User user = db.Users.FirstOrDefault(p => p.Login == auth.Login && p.Password == auth.Password);

if (user == null)

{

ModelState.AddModelError("", "Авторизация неудачна! Проверьте логин и пароль...");

}

else

{

FormsAuthenticationTicket ticket = new FormsAuthenticationTicket(1, user.ID.ToString(), DateTime.Now, DateTime.Now.AddDays(7), true, user.Role.Code);

string strEncrypted = FormsAuthentication.Encrypt(ticket);

Response.Cookies.Add(new HttpCookie("\_\_AUTH", strEncrypted));

return RedirectToAction("Index", "Home");

}

return View(auth);

}

[Authorize]

public ActionResult Logout()

{

FormsAuthentication.SignOut();

return RedirectToAction("Index", "Home");

}

}

}

HomeController.cs

using System.Web.Mvc;

namespace AISLTP.Controllers

{

public class HomeController : Controller

{

public ActionResult Index()

{

return View();

}

}

}

MessagesController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System;

using System.Data.Entity;

using System.Net;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace AISLTP.Controllers

{

public class MessagesController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: Messages

[Authorize(Roles = "admin")]

public async Task<ActionResult> Index()

{

var items = db.Messages.Include(i => i.User).Include(i => i.User.Sotr);

return View(await items.ToListAsync());

}

[Authorize(Roles = "admin")]

public async Task<ActionResult> Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Message item = await db.Messages.FindAsync(id);

if (item == null)

{

return HttpNotFound();

}

if (!item.IsRead)

{

item.IsRead = true;

db.Entry(item).State = EntityState.Modified;

await db.SaveChangesAsync();

}

return View(item);

}

[Authorize]

[HttpPost]

public async Task<ActionResult> Create([Bind(Include = "ID,Date,Content,UserID,IsRead")] Message message)

{

if (ModelState.IsValid)

{

User user = StaticFunctions.GetUser(User.Identity.Name);

if (user != null)

{

message.Date = DateTime.Now;

message.UserID = user.ID;

message.IsRead = false;

db.Messages.Add(message);

await db.SaveChangesAsync();

}

}

return View();

}

}

}

NotifyController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace AISLTP.Controllers

{

public class NotifyController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: Notify

[Authorize]

public ActionResult Index()

{

User user = StaticFunctions.GetUser(User.Identity.Name);

List<Notify> items = new List<Notify>();

DateTime at = DateTime.Today;

DateTime to = at.AddDays(30);

foreach(Lico lico in db.Licos)

{

if (lico.Napravs.Where(p => p.DataOsv >= at && p.DataOsv <= to).Count() > 0)

{

Naprav naprav = lico.Napravs.Where(p => p.DataOsv >= at && p.DataOsv <= to).OrderByDescending(p => p.DataOsv).FirstOrDefault();

if (naprav != null)

{

if (user.RoleID == 1 || naprav.LTP.ID == user.LTP.ID)

{

items.Add(new Notify()

{

Lico = lico,

Date = naprav.DataOsv,

Content = "Предполагаемая дата выхода"

});

}

}

}

}

return View(items);

}

}

}

SearchController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web.Mvc;

namespace AISLTP.Controllers

{

public class SearchController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

private List<Lico> GetList(Search search)

{

List<Lico> items = db.Licos.ToList();

if (search.UseDr)

{

if (search.DrTo < search.DrAt)

{

search.DrTo = search.DrAt;

}

items = items.Where(p => p.Dr >= search.DrAt && p.Dr <= search.DrTo).ToList();

}

if (search.UseDosujd)

{

List<Lico> buffer = new List<Lico>();

if (search.DosujdTo < search.DosujdAt)

{

search.DosujdTo = search.DosujdAt;

}

foreach(Lico lico in items)

{

if (lico.UKs.Where(p => p.DatePrig >= search.DosujdAt && p.DatePrig <= search.DosujdTo).Count() > 0)

{

buffer.Add(lico);

}

}

items = buffer;

}

if (search.UseDosv)

{

List<Lico> buffer = new List<Lico>();

if (search.DosujdTo < search.DosujdAt)

{

search.DosujdTo = search.DosujdAt;

}

foreach (Lico lico in items)

{

if (lico.Osvs.Where(p => p.DateOsv >= search.DosvAt && p.DateOsv <= search.DosvTo).Count() > 0)

{

buffer.Add(lico);

}

}

items = buffer;

}

if (search.UseFIO && search.FIO != null)

{

List<Lico> buffer = new List<Lico>();

foreach (Lico lico in items)

{

string fio = (lico.Fam.Trim() + " " + lico.Ima.Trim() + " " + lico.Otc.Trim()).ToLower();

if (fio.Contains(search.FIO.ToLower()))

{

buffer.Add(lico);

}

}

items = buffer;

}

return items;

}

// GET: Search

[Authorize]

public ActionResult Index()

{

Search search = new Search()

{

UseFIO = false,

FIO = "",

UseDr = false,

DrAt = DateTime.Now,

DrTo = DateTime.Now,

UseDosujd = false,

DosujdAt = DateTime.Now,

DosujdTo = DateTime.Now,

UseDosv = false,

DosvAt = DateTime.Now,

DosvTo = DateTime.Now

};

ViewBag.Items = new List<Lico>();

return View(search);

}

[HttpPost]

[Authorize]

public ActionResult Index(Search search)

{

ViewBag.Items = GetList(search);

return View(search);

}

}

}

UsersController.cs

using System.Data.Entity;

using System.Threading.Tasks;

using System.Net;

using System.Web.Mvc;

using AISLTP.Context;

using AISLTP.Entities;

namespace AISLTP.Controllers

{

public class UsersController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: Users

[Authorize(Roles = "admin")]

public async Task<ActionResult> Index()

{

var users = db.Users.Include(u => u.LTP).Include(u => u.Role).Include(u => u.Sotr);

return View(await users.ToListAsync());

}

// GET: Users/Create

[Authorize(Roles = "admin")]

public ActionResult Create()

{

ViewBag.LtpID = new SelectList(db.LTPs, "ID", "Nom");

ViewBag.RoleID = new SelectList(db.Roles, "ID", "Name");

ViewBag.SotrID = new SelectList(db.Sotrs, "ID", "Cod\_sotr");

return View();

}

// POST: Users/Create

[HttpPost]

[Authorize(Roles = "admin")]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Create([Bind(Include = "ID,Login,Password,SotrID,LtpID,RoleID")] User user)

{

if (ModelState.IsValid)

{

db.Users.Add(user);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

ViewBag.LtpID = new SelectList(db.LTPs, "ID", "Nom", user.LtpID);

ViewBag.RoleID = new SelectList(db.Roles, "ID", "Name", user.RoleID);

ViewBag.SotrID = new SelectList(db.Sotrs, "ID", "Cod\_sotr", user.SotrID);

return View(user);

}

// GET: Users/Edit/5

[Authorize(Roles = "admin")]

public async Task<ActionResult> Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

User user = await db.Users.FindAsync(id);

if (user == null)

{

return HttpNotFound();

}

ViewBag.LtpID = new SelectList(db.LTPs, "ID", "Nom", user.LtpID);

ViewBag.RoleID = new SelectList(db.Roles, "ID", "Name", user.RoleID);

ViewBag.SotrID = new SelectList(db.Sotrs, "ID", "Fio", user.SotrID);

return View(user);

}

// POST: Users/Edit/5

[HttpPost]

[Authorize(Roles = "admin")]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Edit([Bind(Include = "ID,Login,Password,SotrID,LtpID,RoleID")] User user)

{

if (ModelState.IsValid)

{

db.Entry(user).State = EntityState.Modified;

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

ViewBag.LtpID = new SelectList(db.LTPs, "ID", "Nom", user.LtpID);

ViewBag.RoleID = new SelectList(db.Roles, "ID", "Name", user.RoleID);

ViewBag.SotrID = new SelectList(db.Sotrs, "ID", "Cod\_sotr", user.SotrID);

return View(user);

}

// GET: Users/Delete/5

[Authorize(Roles = "admin")]

public async Task<ActionResult> Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

User user = await db.Users.FindAsync(id);

if (user == null)

{

return HttpNotFound();

}

return View(user);

}

// POST: Users/Delete/5

[Authorize(Roles = "admin")]

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DeleteConfirmed(int id)

{

User user = await db.Users.FindAsync(id);

db.Users.Remove(user);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

[Authorize(Roles = "admin")]

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

LTPController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System;

using System.Data.Entity;

using System.Linq;

using System.Web.Mvc;

namespace AISLTP.Controllers.Sp

{

public class LTPController : Controller

{

// GET: LTP

public ActionResult Index()

{

return View();

}

public JsonResult GetLTP(string sidx, string sort, int page, int rows, bool \_search, string searchField, string searchOper, string searchString)

{

ApplicationDbContext db = new ApplicationDbContext();

sort = sort ?? "";

int pageIndex = Convert.ToInt32(page) - 1;

int pageSize = rows;

var LTPList = db.LTPs.Select(

t => new

{

t.ID,

t.Nom,

t.Np,

t.Ul,

t.Dom,

t.Pindex,

t.Teldej,

t.Email

});

if (\_search)

{

switch (searchField)

{

case "Nom":

LTPList = LTPList.Where(t => t.Nom.Contains(searchString));

break;

}

}

int totalRecords = LTPList.Count();

var totalPages = (int)Math.Ceiling((float)totalRecords / (float)rows);

if (sort.ToUpper() == "DESC")

{

LTPList = LTPList.OrderByDescending(t => t.Nom);

LTPList = LTPList.Skip(pageIndex \* pageSize).Take(pageSize);

}

else

{

LTPList = LTPList.OrderBy(t => t.Nom);

LTPList = LTPList.Skip(pageIndex \* pageSize).Take(pageSize);

}

var jsonData = new

{

total = totalPages,

page,

records = totalRecords,

rows = LTPList

};

return Json(jsonData, JsonRequestBehavior.AllowGet);

}

[HttpPost]

public string Create([Bind(Exclude = "Id")] LTP Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.LTPs.Add(Model);

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Edit(LTP Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.Entry(Model).State = EntityState.Modified;

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Delete(int Id)

{

ApplicationDbContext db = new ApplicationDbContext();

LTP lTPs = db.LTPs.Find(Id);

db.LTPs.Remove(lTPs);

db.SaveChanges();

return "Удалено успешно";

}

}

}

MedcomController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System;

using System.Data.Entity;

using System.Linq;

using System.Web.Mvc;

namespace AISLTP.Controllers.Sp

{

public class MedcomController : Controller

{

// GET: Med

public ActionResult Index()

{

return View();

}

public JsonResult GetMedcom(string sidx, string sort, int page, int rows, bool \_search, string searchField, string searchOper, string searchString)

{

ApplicationDbContext db = new ApplicationDbContext();

sort = sort ?? "";

int pageIndex = Convert.ToInt32(page) - 1;

int pageSize = rows;

var MedcomList = db.Medcoms.Select(

t => new

{

t.ID,

t.Txt,

});

if (\_search)

{

switch (searchField)

{

case "Txt":

MedcomList = MedcomList.Where(t => t.Txt.Contains(searchString));

break;

}

}

int totalRecords = MedcomList.Count();

var totalPages = (int)Math.Ceiling((float)totalRecords / (float)rows);

if (sort.ToUpper() == "DESC")

{

MedcomList = MedcomList.OrderByDescending(t => t.Txt);

MedcomList = MedcomList.Skip(pageIndex \* pageSize).Take(pageSize);

}

else

{

MedcomList = MedcomList.OrderBy(t => t.Txt);

MedcomList = MedcomList.Skip(pageIndex \* pageSize).Take(pageSize);

}

var jsonData = new

{

total = totalPages,

page,

records = totalRecords,

rows = MedcomList

};

return Json(jsonData, JsonRequestBehavior.AllowGet);

}

[HttpPost]

public string Create([Bind(Exclude = "Id")] Medcom Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.Medcoms.Add(Model);

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Edit(Medcom Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.Entry(Model).State = EntityState.Modified;

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Delete(int Id)

{

ApplicationDbContext db = new ApplicationDbContext();

Medcom Medcoms = db.Medcoms.Find(Id);

db.Medcoms.Remove(Medcoms);

db.SaveChanges();

return "Удалено успешно";

}

}

}

OsnnapController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System;

using System.Data.Entity;

using System.Linq;

using System.Web.Mvc;

namespace AISLTP.Controllers.Sp

{

public class OsnnapController : Controller

{

// GET: Osnnap

public ActionResult Index()

{

return View();

}

public JsonResult GetOsnnap(string sidx, string sort, int page, int rows, bool \_search, string searchField, string searchOper, string searchString)

{

ApplicationDbContext db = new ApplicationDbContext();

sort = sort ?? "";

int pageIndex = Convert.ToInt32(page) - 1;

int pageSize = rows;

var OsnnapList = db.Osnnaps.Select(

t => new

{

t.ID,

t.Txt,

});

if (\_search)

{

switch (searchField)

{

case "Txt":

OsnnapList = OsnnapList.Where(t => t.Txt.Contains(searchString));

break;

}

}

int totalRecords = OsnnapList.Count();

var totalPages = (int)Math.Ceiling((float)totalRecords / (float)rows);

if (sort.ToUpper() == "DESC")

{

OsnnapList = OsnnapList.OrderByDescending(t => t.Txt);

OsnnapList = OsnnapList.Skip(pageIndex \* pageSize).Take(pageSize);

}

else

{

OsnnapList = OsnnapList.OrderBy(t => t.Txt);

OsnnapList = OsnnapList.Skip(pageIndex \* pageSize).Take(pageSize);

}

var jsonData = new

{

total = totalPages,

page,

records = totalRecords,

rows = OsnnapList

};

return Json(jsonData, JsonRequestBehavior.AllowGet);

}

[HttpPost]

public string Create([Bind(Exclude = "Id")] Osnnap Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.Osnnaps.Add(Model);

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Edit(Osnnap Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.Entry(Model).State = EntityState.Modified;

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Delete(int Id)

{

ApplicationDbContext db = new ApplicationDbContext();

Osnnap Osnnaps = db.Osnnaps.Find(Id);

db.Osnnaps.Remove(Osnnaps);

db.SaveChanges();

return "Удалено успешно";

}

}

}

ProfController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System;

using System.Data.Entity;

using System.Linq;

using System.Web.Mvc;

namespace AISLTP.Controllers.Sp

{

public class ProfController : Controller

{

// GET: Prof

public ActionResult Index()

{

return View();

}

public JsonResult GetProf(string sidx, string sort, int page, int rows, bool \_search, string searchField, string searchOper, string searchString)

{

ApplicationDbContext db = new ApplicationDbContext();

sort = sort ?? "";

int pageIndex = Convert.ToInt32(page) - 1;

int pageSize = rows;

var ProfList = db.Profs.Select(

t => new

{

t.ID,

t.Txt,

});

if (\_search)

{

switch (searchField)

{

case "Txt":

ProfList = ProfList.Where(t => t.Txt.Contains(searchString));

break;

}

}

int totalRecords = ProfList.Count();

var totalPages = (int)Math.Ceiling((float)totalRecords / (float)rows);

if (sort.ToUpper() == "DESC")

{

ProfList = ProfList.OrderByDescending(t => t.Txt);

ProfList = ProfList.Skip(pageIndex \* pageSize).Take(pageSize);

}

else

{

ProfList = ProfList.OrderBy(t => t.Txt);

ProfList = ProfList.Skip(pageIndex \* pageSize).Take(pageSize);

}

var jsonData = new

{

total = totalPages,

page,

records = totalRecords,

rows = ProfList

};

return Json(jsonData, JsonRequestBehavior.AllowGet);

}

[HttpPost]

public string Create([Bind(Exclude = "Id")] Prof Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.Profs.Add(Model);

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Edit(Prof Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.Entry(Model).State = EntityState.Modified;

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Delete(int Id)

{

ApplicationDbContext db = new ApplicationDbContext();

Prof Profs = db.Profs.Find(Id);

db.Profs.Remove(Profs);

db.SaveChanges();

return "Удалено успешно";

}

}

}

SotrController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System;

using System.Data.Entity;

using System.Linq;

using System.Web.Mvc;

namespace AISLTP.Controllers.Sp

{

public class SotrController : Controller

{

//

// GET: /Sotr/

public ActionResult Index()

{

return View();

}

public JsonResult GetSotr(string sidx, string sort, int page, int rows, bool \_search, string searchField, string searchOper, string searchString)

{

ApplicationDbContext db = new ApplicationDbContext();

sort = sort ?? "";

int pageIndex = Convert.ToInt32(page) - 1;

int pageSize = rows;

var SotrList = db.Sotrs.Select(

t => new

{

t.ID,

t.Cod\_sotr,

t.Ima,

t.Fio,

t.Otc,

t.Dr,

t.Sex,

t.Dvi

});

if (\_search)

{

switch (searchField)

{

case "Cod\_sotr":

SotrList = SotrList.Where(t => t.Cod\_sotr.Contains(searchString));

break;

case "Ima":

SotrList = SotrList.Where(t => t.Ima.Contains(searchString));

break;

case "Fio":

SotrList = SotrList.Where(t => t.Fio.Contains(searchString));

break;

case "Otc":

SotrList = SotrList.Where(t => t.Otc.Contains(searchString));

break;

}

}

int totalRecords = SotrList.Count();

var totalPages = (int)Math.Ceiling((float)totalRecords / (float)rows);

if (sort.ToUpper() == "DESC")

{

SotrList = SotrList.OrderByDescending(t => t.Ima);

SotrList = SotrList.Skip(pageIndex \* pageSize).Take(pageSize);

}

else

{

SotrList = SotrList.OrderBy(t => t.Fio);

SotrList = SotrList.Skip(pageIndex \* pageSize).Take(pageSize);

}

var jsonData = new

{

total = totalPages,

page,

records = totalRecords,

rows = SotrList

};

return Json(jsonData, JsonRequestBehavior.AllowGet);

}

[HttpPost]

public string Create([Bind(Exclude = "Id")] Sotr Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.Sotrs.Add(Model);

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Edit(Sotr Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.Entry(Model).State = EntityState.Modified;

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Delete(int Id)

{

ApplicationDbContext db = new ApplicationDbContext();

Sotr Sotr = db.Sotrs.Find(Id);

db.Sotrs.Remove(Sotr);

db.SaveChanges();

return "Удалено успешно";

}

}

}

UKController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System;

using System.Data.Entity;

using System.Linq;

using System.Web.Mvc;

namespace AISLTP.Controllers.Sp

{

public class UKController : Controller

{

// GET: UK

public ActionResult Index()

{

return View();

}

public JsonResult GetUK(string sidx, string sort, int page, int rows, bool \_search, string searchField, string searchOper, string searchString)

{

ApplicationDbContext db = new ApplicationDbContext();

sort = sort ?? "";

int pageIndex = Convert.ToInt32(page) - 1;

int pageSize = rows;

var UKList = db.UKs.Select(

t => new

{

t.ID,

t.Point,

t.Part,

t.Article,

t.Name,

t.Prim,

});

if (\_search)

{

switch (searchField)

{

case "Nom":

UKList = UKList.Where(t => t.Name.Contains(searchString));

break;

}

}

int totalRecords = UKList.Count();

var totalPages = (int)Math.Ceiling((float)totalRecords / (float)rows);

if (sort.ToUpper() == "DESC")

{

UKList = UKList.OrderByDescending(t => t.Name);

UKList = UKList.Skip(pageIndex \* pageSize).Take(pageSize);

}

else

{

UKList = UKList.OrderBy(t => t.Name);

UKList = UKList.Skip(pageIndex \* pageSize).Take(pageSize);

}

var jsonData = new

{

total = totalPages,

page,

records = totalRecords,

rows = UKList

};

return Json(jsonData, JsonRequestBehavior.AllowGet);

}

[HttpPost]

public string Create([Bind(Exclude = "Id")] UK Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.UKs.Add(Model);

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Edit(UK Model)

{

ApplicationDbContext db = new ApplicationDbContext();

string msg;

try

{

if (ModelState.IsValid)

{

db.Entry(Model).State = EntityState.Modified;

db.SaveChanges();

msg = "Сохранено успешно";

}

else

{

msg = "Данные не прошли проверку ввода";

}

}

catch (Exception ex)

{

msg = "Произошла ошибка:" + ex.Message;

}

return msg;

}

public string Delete(int Id)

{

ApplicationDbContext db = new ApplicationDbContext();

UK UKs = db.UKs.Find(Id);

db.UKs.Remove(UKs);

db.SaveChanges();

return "Удалено успешно";

}

}

}

AddressController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System.Data.Entity;

using System.Net;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace AISLTP.Controllers.Journals\_registrations

{

public class AddressController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: Address

[Authorize]

public async Task<ActionResult> Index()

{

var licos = db.Licos.Include(l => l.Nac).Include(l => l.Np).Include(l => l.Obl).Include(l => l.Pol).Include(l => l.Rn);

return View(await licos.ToListAsync());

}

// GET: Address/Show/5

[Authorize]

public async Task<ActionResult> Show(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Lico lico = await db.Licos.FindAsync(id);

if (lico == null)

{

return HttpNotFound();

}

ViewBag.Fam = lico.Fam;

ViewBag.Ima = lico.Ima;

ViewBag.Otc = lico.Otc;

Session["IDLico"] = lico.ID;

return View(lico);

}

// POST: Address/Show/5

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Show([Bind(Include = "ID,Fam,Ima,Otc,Dr,PolID,Pasport,NacID,OblID,RnID,NpID,Vneshnost,Prim")] Lico lico)

{

if (ModelState.IsValid)

{

db.Entry(lico).State = EntityState.Modified;

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

ViewBag.NacID = new SelectList(db.Nacs, "ID", "Txt", lico.NacID);

ViewBag.NpID = new SelectList(db.Nps, "ID", "Txt", lico.NpID);

ViewBag.OblID = new SelectList(db.Obls, "ID", "Txt", lico.OblID);

ViewBag.PolID = new SelectList(db.Pols, "ID", "Txt", lico.PolID);

ViewBag.RnID = new SelectList(db.Rns, "ID", "Txt", lico.RnID);

return View(lico);

}

[Authorize]

public ActionResult CreateAddress()

{

ViewBag.NpID = new SelectList(db.Nps, "ID", "Txt");

ViewBag.OblID = new SelectList(db.Obls, "ID", "Txt");

ViewBag.RnID = new SelectList(db.Rns, "ID", "Txt");

return View();

}

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> CreateAddress([Bind(Include = "ID,OblID,RnID,NpID,Ul,Dom,Korpus,Kvartira,Prim")] Address CreateAddress)

{

if (ModelState.IsValid)

{

db.Licos.Find(Session["IDLico"]).Addresses.Add(CreateAddress);

//db.Addresses.Add(CreateAddress);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

ViewBag.NpID = new SelectList(db.Nps, "ID", "Txt", CreateAddress.NpID);

ViewBag.OblID = new SelectList(db.Obls, "ID", "Txt", CreateAddress.OblID);

ViewBag.RnID = new SelectList(db.Rns, "ID", "Txt", CreateAddress.RnID);

return View(CreateAddress);

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

EducsController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace AISLTP.Controllers.Journals\_registrations

{

public class EducsController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: Educs

[Authorize]

public async Task<ActionResult> Index()

{

var licos = db.Licos.Include(l => l.Nac).Include(l => l.Np).

Include(l => l.Obl).Include(l => l.Pol).Include(l => l.Rn).Include(l => l.Addresses);

return View(await licos.ToListAsync());

}

// GET: Educs/Details/5

[Authorize]

public ActionResult Details(int id = 0)

{

Lico lico = db.Licos.Find(id);

if (lico == null)

{

return HttpNotFound();

}

return View(lico);

}

[Authorize]

public ActionResult Edit(int id = 0)

{

Lico lico = db.Licos.Find(id);

if (lico == null)

{

return HttpNotFound();

}

ViewBag.Educs = db.Educs.ToList();

return View(lico);

}

[HttpPost]

[Authorize]

public ActionResult Edit(Lico lico, int[] selectedEducs)

{

Lico newLico = db.Licos.Find(lico.ID);

newLico.Fam = lico.Fam;

newLico.Ima = lico.Ima;

newLico.Otc = lico.Otc;

newLico.Educs.Clear();

if (selectedEducs != null)

{

//получаем выбранные курсы

foreach (var c in db.Educs.Where(co => selectedEducs.Contains(co.ID)))

{

newLico.Educs.Add(c);

}

}

db.Entry(newLico).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Details", new { Id = newLico.ID });

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

LicoesController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System.Data.Entity;

using System.IO;

using System.Linq;

using System.Net;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

namespace AISLTP.Controllers.Journals\_registrations

{

public class LicoesController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: Licoes

[Authorize]

public async Task<ActionResult> Index()

{

var licos = db.Licos.Include(l => l.Nac).Include(l => l.Np).

Include(l => l.Obl).Include(l => l.Pol).Include(l => l.Rn).Include(l => l.Addresses);

//var addresses = db.Addresses.Include( a => a.Np ).Include( a => a.Obl ).Include( a => a.Rn );

return View(await licos.ToListAsync());

}

// GET: Licoes/Details/5

[Authorize]

public async Task<ActionResult> Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Lico lico = await db.Licos.FindAsync(id);

if (lico == null)

{

return HttpNotFound();

}

return View(lico);

}

// GET: Licoes/Create

[Authorize]

public ActionResult Create()

{

ViewBag.NacID = new SelectList(db.Nacs, "ID", "Txt");

ViewBag.NpID = new SelectList(db.Nps, "ID", "Txt");

ViewBag.OblID = new SelectList(db.Obls, "ID", "Txt");

ViewBag.PolID = new SelectList(db.Pols, "ID", "Txt");

ViewBag.RnID = new SelectList(db.Rns, "ID", "Txt");

return View();

}

// POST: Licoes/Create

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Create([Bind(Include = "ID,Fam,Ima,Otc,Dr,PolID,Pasport,NacID,OblID,RnID,NpID,Vneshnost")] Lico lico, HttpPostedFileBase uploadFoto)

{

if (ModelState.IsValid)

{

if (uploadFoto != null)

{

byte[] fotoData = null;

using (var binaryReader = new BinaryReader(uploadFoto.InputStream))

{

fotoData = binaryReader.ReadBytes(uploadFoto.ContentLength);

}

lico.Foto = fotoData;

}

db.Licos.Add(lico);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

ViewBag.NacID = new SelectList(db.Nacs, "ID", "Txt", lico.NacID);

ViewBag.NpID = new SelectList(db.Nps, "ID", "Txt", lico.NpID);

ViewBag.OblID = new SelectList(db.Obls, "ID", "Txt", lico.OblID);

ViewBag.PolID = new SelectList(db.Pols, "ID", "Txt", lico.PolID);

ViewBag.RnID = new SelectList(db.Rns, "ID", "Txt", lico.RnID);

return View(lico);

}

// GET: Licoes/Edit/5

[Authorize]

public async Task<ActionResult> Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Lico lico = await db.Licos.FindAsync(id);

if (lico == null)

{

return HttpNotFound();

}

ViewBag.NacID = new SelectList(db.Nacs, "ID", "Txt", lico.NacID);

ViewBag.NpID = new SelectList(db.Nps, "ID", "Txt", lico.NpID);

ViewBag.OblID = new SelectList(db.Obls, "ID", "Txt", lico.OblID);

ViewBag.PolID = new SelectList(db.Pols, "ID", "Txt", lico.PolID);

ViewBag.RnID = new SelectList(db.Rns, "ID", "Txt", lico.RnID);

ViewBag.Address = db.Addresses.ToList();

return View(lico);

}

// POST: Licoes/Edit/5

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Edit([Bind(Include = "ID,Fam,Ima,Otc,Dr,PolID,Pasport,NacID,OblID,RnID,NpID,Vneshnost,Foto")] Lico lico, HttpPostedFileBase uploadFoto)

{

if (ModelState.IsValid)

{

if (uploadFoto != null)

{

byte[] fotoData = null;

using (var binaryReader = new BinaryReader(uploadFoto.InputStream))

{

fotoData = binaryReader.ReadBytes(uploadFoto.ContentLength);

}

lico.Foto = fotoData;

}

db.Entry(lico).State = EntityState.Modified;

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

ViewBag.NacID = new SelectList(db.Nacs, "ID", "Txt", lico.NacID);

ViewBag.NpID = new SelectList(db.Nps, "ID", "Txt", lico.NpID);

ViewBag.OblID = new SelectList(db.Obls, "ID", "Txt", lico.OblID);

ViewBag.PolID = new SelectList(db.Pols, "ID", "Txt", lico.PolID);

ViewBag.RnID = new SelectList(db.Rns, "ID", "Txt", lico.RnID);

return View(lico);

}

// GET: Licoes/Delete/5

[Authorize]

public async Task<ActionResult> Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Lico lico = await db.Licos.FindAsync(id);

if (lico == null)

{

return HttpNotFound();

}

return View(lico);

}

// POST: Licoes/Delete/5

[HttpPost, ActionName("Delete")]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DeleteConfirmed(int id)

{

Lico lico = await db.Licos.FindAsync(id);

db.Licos.Remove(lico);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

JournalLTPController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System.Data.Entity;

using System.Net;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace AISLTP.Controllers.Journals\_registrations.JournalLTP

{

public class JournalLTPController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: JournalLTP

[Authorize]

public async Task<ActionResult> Index()

{

var licos = db.Licos.Include(l => l.Nac).Include(l => l.Np).Include(l => l.Obl).Include(l => l.Pol).Include(l => l.Rn);

return View(await licos.ToListAsync());

}

[Authorize]

public async Task<ActionResult> Show(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Lico lico = await db.Licos.FindAsync(id);

if (lico == null)

{

return HttpNotFound();

}

ViewBag.Fam = lico.Fam;

ViewBag.Ima = lico.Ima;

ViewBag.Otc = lico.Otc;

Session["IDLico"] = lico.ID;

return View(lico);

}

}

}

NapravsController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System.Data.Entity;

using System.Net;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace AISLTP.Controllers.Journals\_registrations.JournalLTP

{

public class NapravsController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

[Authorize]

public async Task<ActionResult> Index()

{

var licos = db.Licos.Include(l => l.Nac).Include(l => l.Np).Include(l => l.Obl).Include(l => l.Pol).Include(l => l.Rn);

return View(await licos.ToListAsync());

//var medics = db.Medics.Include(m => m.Medcom);

//return View(await medics.ToListAsync());

}

[Authorize]

public async Task<ActionResult> Show(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Lico lico = await db.Licos.FindAsync(id);

if (lico == null)

{

return HttpNotFound();

}

ViewBag.Fam = lico.Fam;

ViewBag.Ima = lico.Ima;

ViewBag.Otc = lico.Otc;

Session["IDLico"] = lico.ID;

return View(lico);

}

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Show([Bind(Include = "ID,OsnnapID,LTPID,DataPom,DataOsv,FamD,ImaD,OtcD,TelD,FamO,ImaO,OtcO,TelO")] Lico lico)

{

if (ModelState.IsValid)

{

db.Entry(lico).State = EntityState.Modified;

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(lico);

}

[Authorize]

public ActionResult CreateNaprav()

{

ViewBag.OsnnapID = new SelectList(db.Osnnaps, "ID", "Txt");

ViewBag.LTPID = new SelectList(db.LTPs, "ID", "Nom");

return View();

}

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> CreateNaprav([Bind(Include = "ID,OsnnapID,LTPID,DataPom,DataOsv,FamD,ImaD,OtcD,TelD,FamO,ImaO,OtcO,TelO")] Naprav CreateNaprav)

{

if (ModelState.IsValid)

{

db.Licos.Find(Session["IDLico"]).Napravs.Add(CreateNaprav);

//db.Addresses.Add(CreateAddress);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

ViewBag.OsnnapID = new SelectList(db.Osnnaps, "ID", "Txt", CreateNaprav.OsnnapID);

ViewBag.LTPID = new SelectList(db.LTPs, "ID", "Nom", CreateNaprav.LTPID);

return View(CreateNaprav);

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

OsvsController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System.Data.Entity;

using System.Net;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace AISLTP.Controllers.Journals\_registrations.JournalLTP

{

public class OsvsController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: Osvs

[Authorize]

public async Task<ActionResult> Index()

{

var licos = db.Licos.Include(l => l.Nac).Include(l => l.Np).Include(l => l.Obl).Include(l => l.Pol).Include(l => l.Rn);

return View(await licos.ToListAsync());

}

[Authorize]

public async Task<ActionResult> Show(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Lico lico = await db.Licos.FindAsync(id);

if (lico == null)

{

return HttpNotFound();

}

ViewBag.Fam = lico.Fam;

ViewBag.Ima = lico.Ima;

ViewBag.Otc = lico.Otc;

Session["IDLico"] = lico.ID;

return View(lico);

}

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Show([Bind(Include = "ID,DateOsv,DatePrib,Control,Locat,Dost,Trud,Posic,Bron")] Lico lico)

{

if (ModelState.IsValid)

{

db.Entry(lico).State = EntityState.Modified;

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(lico);

}

[Authorize]

public ActionResult CreateOsv()

{

return View();

}

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> CreateOsv([Bind(Include = "ID,DateOsv,DatePrib,Control,Locat,Dost,Trud,Posic,Bron")] Osv CreateOsv)

{

if (ModelState.IsValid)

{

db.Licos.Find(Session["IDLico"]).Osvs.Add(CreateOsv);

//db.Addresses.Add(CreateAddress);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(CreateOsv);

}

// GET: Osvs/Details/5

[Authorize]

public async Task<ActionResult> Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Osv osv = await db.Osvs.FindAsync(id);

if (osv == null)

{

return HttpNotFound();

}

return View(osv);

}

// GET: Osvs/Create

[Authorize]

public ActionResult Create()

{

return View();

}

// POST: Osvs/Create

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Create([Bind(Include = "ID,DateOsv,DatePrib,Control,Locat,Dost,Trud,Posic,Bron")] Osv osv)

{

if (ModelState.IsValid)

{

db.Osvs.Add(osv);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(osv);

}

// GET: Osvs/Edit/5

[Authorize]

public async Task<ActionResult> Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Osv osv = await db.Osvs.FindAsync(id);

if (osv == null)

{

return HttpNotFound();

}

return View(osv);

}

// POST: Osvs/Edit/5

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Edit([Bind(Include = "ID,DateOsv,DatePrib,Control,Locat,Dost,Trud,Posic,Bron")] Osv osv)

{

if (ModelState.IsValid)

{

db.Entry(osv).State = EntityState.Modified;

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(osv);

}

// GET: Osvs/Delete/5

[Authorize]

public async Task<ActionResult> Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Osv osv = await db.Osvs.FindAsync(id);

if (osv == null)

{

return HttpNotFound();

}

return View(osv);

}

// POST: Osvs/Delete/5

[HttpPost, ActionName("Delete")]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DeleteConfirmed(int id)

{

Osv osv = await db.Osvs.FindAsync(id);

db.Osvs.Remove(osv);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

WorkLTPsController.cs

using AISLTP.Context;

using AISLTP.Entities;

using System.Data.Entity;

using System.Net;

using System.Threading.Tasks;

using System.Web.Mvc;

namespace AISLTP.Controllers.Journals\_registrations.JournalLTP

{

public class WorkLTPsController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: WorkLTPs

[Authorize]

public async Task<ActionResult> Index()

{

var licos = db.Licos.Include(l => l.Nac).Include(l => l.Np).Include(l => l.Obl).Include(l => l.Pol).Include(l => l.Rn);

return View(await licos.ToListAsync());

}

[Authorize]

public async Task<ActionResult> Show(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Lico lico = await db.Licos.FindAsync(id);

if (lico == null)

{

return HttpNotFound();

}

ViewBag.Fam = lico.Fam;

ViewBag.Ima = lico.Ima;

ViewBag.Otc = lico.Otc;

Session["IDLico"] = lico.ID;

return View(lico);

}

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Show([Bind(Include = "ID,Otn,Trud")] Lico lico)

{

if (ModelState.IsValid)

{

db.Entry(lico).State = EntityState.Modified;

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(lico);

}

[Authorize]

public ActionResult CreateWorkLTP()

{

return View();

}

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<ActionResult> CreateWorkLTP([Bind(Include = "ID,Otn,Trud")] WorkLTP CreateWorkLTP)

{

if (ModelState.IsValid)

{

db.Licos.Find(Session["IDLico"]).WorkLTPs.Add(CreateWorkLTP);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(CreateWorkLTP);

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

IdentityContext.cs

using AISLTP.Entities;

using System;

using System.Data.Entity;

namespace AISLTP.Context

{

public class ApplicationDbContext : DbContext

{

static ApplicationDbContext()

{

Database.SetInitializer<ApplicationDbContext>(new MyContextInitializer());

}

public ApplicationDbContext() : base("DefaultConnection")

{ }

public DbSet<Sotr> Sotrs { get; set; }

public DbSet<Lico> Licos { get; set; }

public DbSet<Pol> Pols { get; set; }

public DbSet<Nac> Nacs { get; set; }

public DbSet<Obl> Obls { get; set; }

public DbSet<Rn> Rns { get; set; }

public DbSet<Np> Nps { get; set; }

public DbSet<Address> Addresses { get; set; }

public DbSet<LTP> LTPs { get; set; }

public DbSet<Medcom> Medcoms { get; set; }

public DbSet<Osnnap> Osnnaps { get; set; }

public DbSet<Doc> Docs { get; set; }

public DbSet<Educ> Educs { get; set; }

public DbSet<Prof> Profs { get; set; }

public DbSet<Spec> Specs { get; set; }

public DbSet<Court> Courts { get; set; }

public DbSet<Vidsv> Vidsvs { get; set; }

public DbSet<Viddebt> Viddebts { get; set; }

public DbSet<Otnosh> Otnoshs { get; set; }

public DbSet<Osnsocr> Osnsocrs { get; set; }

public DbSet<Osnprod> Osnprods { get; set; }

public DbSet<Osndosr> Osndosrs { get; set; }

public DbSet<Samovol> Samovols { get; set; }

public DbSet<UK> UKs { get; set; }

public DbSet<Koap> Koaps { get; set; }

public DbSet<Medic> Medics { get; set; }

public DbSet<Privent> Privents { get; set; }

public DbSet<Naprav> Napravs { get; set; }

public DbSet<Objal> Objals { get; set; }

public DbSet<Otrad> Otrads { get; set; }

public DbSet<Debt> Debts { get; set; }

public DbSet<Svaz> Svazs { get; set; }

public DbSet<Work> Works { get; set; }

public DbSet<WorkLTP> WorkLTPs { get; set; }

public DbSet<Lechen> Lechens { get; set; }

public DbSet<Zavis> Zaviss { get; set; }

public DbSet<Obchest> Obchests { get; set; }

public DbSet<Soderotnosh> Soderotnoshes { get; set; }

public DbSet<Psix> Psixes { get; set; }

public DbSet<Discip> Discips { get; set; }

public DbSet<Samohod> Samohods { get; set; }

public DbSet<Osv> Osvs { get; set; }

public DbSet<User> Users { get; set; }

public DbSet<Role> Roles { get; set; }

public DbSet<Message> Messages { get; set; }

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Entity<Address>().HasMany(c => c.Licos)

.WithMany(s => s.Addresses)

.Map(t => t.MapLeftKey("AddressID")

.MapRightKey("LicoID")

.ToTable("Address\_Lico"));

modelBuilder.Entity<Prof>().HasMany(c => c.Licos)

.WithMany(s => s.Profs)

.Map(t => t.MapLeftKey("ProfId")

.MapRightKey("LicoId")

.ToTable("ProfLico"));

modelBuilder.Entity<Lico>().Property(p => p.Foto)

.HasColumnType("image");

}

}

class MyContextInitializer : DropCreateDatabaseIfModelChanges<ApplicationDbContext>

{

// Инициализация данных по-умолчанию

protected override void Seed(ApplicationDbContext context)

{

// Роли

context.Roles.Add(new Role() { ID = 1, Name = "Администратор", Code = "admin" });

context.Roles.Add(new Role() { ID = 2, Name = "Сотрудник", Code = "simple" });

// Администратор

context.Users.Add(new User() { Login = "sa", Password = "1", RoleID = 1 });

// Пол

context.Pols.Add(new Pol() { Txt = "Мужской" });

context.Pols.Add(new Pol() { Txt = "Женский" });

// ЛТП

context.LTPs.Add(new LTP()

{

Nom = "ЛТП №6",

Np = "Дзержинск",

Ul = "пер. Сервеный",

Dom = "1",

Pindex = 222720,

Teldej = "8(01716)40-8-45",

Email = "dpltp6@mail.ru"

});

context.LTPs.Add(new LTP()

{

Nom = "ЛТП №9",

Np = "Витебск",

Ul = "ул. Гагарина",

Dom = "48",

Pindex = 210026,

Teldej = "8(0212)54-89-60",

Email = "dpltp9@mail.ru"

});

// Сохранить изменения

context.SaveChanges();

}

}

}

\_Layout.cshtml

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewBag.Title</title>

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

</head>

<body>

<nav class="navbar navbar-inverse navbar-fixed-top">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

@Html.ActionLink("АИС-ЛТП", "Index", "Home", new { }, new { @class = "navbar-brand" })

</div>

<div class="collapse navbar-collapse">

@if (User.Identity.IsAuthenticated)

{

<ul class="nav navbar-nav">

<li>@Html.ActionLink("Поиск", "Index", "Search")</li>

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown">Журналы учета <b class="caret"></b></a>

<ul class="dropdown-menu">

<li>@Html.ActionLink("Журнал лиц", "Index", "Licoes") </li>

<li>@Html.ActionLink("Журнал адресов", "Index", "Address") </li>

<li class="divider"></li>

<li>@Html.ActionLink("Журнал осуждений", "Index", "UKs") </li>

<li>@Html.ActionLink("Журнал правонарушений", "Index", "Koaps") </li>

<li>@Html.ActionLink("Журнал образований", "Index", "Educs")</li>

<li>@Html.ActionLink("Журнал профессий", "Index", "Profs")</li>

<li>@Html.ActionLink("Журнал специальностей", "Index", "Specs")</li>

<li>@Html.ActionLink("Журнал медицинских комиссий", "Index", "Medics")</li>

<li>@Html.ActionLink("Журнал превентивных надзоров", "Index", "Privents")</li>

<li class="divider"></li>

<li>@Html.ActionLink("Журнал содержания ЛТП", "Index", "JournalLTP")</li>

</ul>

</li>

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown">Вспомогательные журналы<b class="caret"></b></a>

<ul class="dropdown-menu">

<li>@Html.ActionLink("Справочники", "Index", "Sp")</li>

@if (User.IsInRole("admin"))

{

<li>@Html.ActionLink("Сообщения", "Index", "Messages")</li>

<li>@Html.ActionLink("Пользователи", "Index", "Users")</li>

}

</ul>

</li>

</ul>

<ul class="nav navbar-nav navbar-right">

<li>@Html.ActionLink("Выйти", "Logout", "Account")</li>

<li><a href="~/Notify" title="Уведомления"><span class="glyphicon glyphicon-warning-sign"></span></a></li>

</ul>

AISLTP.Entities.User user = AISLTP.StaticFunctions.GetUser(User.Identity.Name);

string userName = user.Login;

if (user.Sotr != null)

{

userName = user.Sotr.Fio;

}

<p class="navbar-text navbar-right">[@userName]</p>

}

else

{

<ul class="nav navbar-nav navbar-right">

<li>@Html.ActionLink("Войти", "Index", "Account")</li>

</ul>

}

</div>

</nav>

<div class="container body-content">

@RenderBody()

<hr />

<footer>

<p>

&copy; @DateTime.Now.Year

</p>

</footer>

</div>

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/bootstrap")

@RenderSection("scripts", required: false)

@if (User.Identity.IsAuthenticated)

{

<button type="button" class="btn btn-danger" id="modalMessageButton" data-toggle="modal" data-target="#modalMessage" title="Сообщение администратору">

<span class="glyphicon glyphicon-bullhorn"></span>

</button>

<div class="modal fade" id="modalMessage" tabindex="-1" role="dialog" aria-labelledby="modalMessageLabel">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

<h4 class="modal-title" id="modalMessageLabel">Сообщение администратору</h4>

</div>

<div class="modal-body">

<div class="form-group">

<label for="modalMessageContent">Содержание:</label>

<textarea class="form-control" id="modalMessageContent" style="max-width: initial; min-height: 100px;"></textarea>

<p class="help-block text-danger">Укажите текст сообщения</p>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-primary" data-dismiss="modal">

<span class="glyphicon glyphicon-remove"></span>

</button>

<button type="button" class="btn btn-primary" id="modalMessageSend">

<span class="glyphicon glyphicon-send"></span>

</button>

</div>

</div>

</div>

</div>

@Scripts.Render("~/bundles/mmessage");

}

</body>

</html>

Account/Index.cshtml

@model AISLTP.Entities.Auth

@{

ViewBag.Title = "Авторизация";

Layout = "~/Views/Shared/\_Auth.cshtml";

}

<div class="row">

<div class="col-md-6 col-md-offset-3">

<h1>Авторизация</h1>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

@Html.ValidationSummary(true, "", new { @class = "alert alert-warning" })

<div class="form-group">

@Html.LabelFor(model => model.Login)

@Html.TextBoxFor(model => model.Login, new { @class = "form-control", style = "max-width: initial;" })

<small class="form-text text-danger">

@Html.ValidationMessageFor(model => model.Login)

</small>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Password)

@Html.PasswordFor(model => model.Password, new { @class = "form-control", style = "max-width: initial;" })

<small class="form-text text-danger">

@Html.ValidationMessageFor(model => model.Password)

</small>

</div>

<input type="submit" value="Авторизация" class="btn btn-primary">

}

</div>

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

Licoes/Details.cshtml

@using AISLTP.Entities

@model Lico

@{

ViewBag.Title = "Информация о лицу";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Информация о лице (<b>@Model.Fam @Model.Ima @Model.Otc</b>):</h2>

<div>

<hr />

<dl class="dl-horizontal">

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingTwentyFour">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseTwentyFour" aria-expanded="false" aria-controls="collapseTwentyFour">

Рекомендации:

</a>

</h4>

</div>

<div id="collapseTwentyFour" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTwentyFour">

<div class="panel-body">

<ul>

<li>Не направлять данное лицо на работы за пределы ЛТП в целях недопущения рецедива самовольного оставления ЛТП</li>

<li>Направить его в Новогрудское ЛТП для наркоманов</li>

<li>Направить его в отряд специализирующийся на медико-социальной реадаптации судимых лиц</li>

<li>Недопускать предельное содержание в отряде таких лиц с неоднократным совершением дисциплинарных нарушений правил внутреннего режима ЛТП или административного правонарушения во время содержания в ЛТП</li>

</ul>

</div>

</div>

</div>

<div class="panel-group" id="accordion" role="tablist" aria-multiselectable="true">

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingOne">

<h4 class="panel-title">

<a role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseOne" aria-expanded="true" aria-controls="collapseOne">

Персональные данные

</a>

</h4>

</div>

<div id="collapseOne" class="panel-collapse collapse in" role="tabpanel" aria-labelledby="headingOne">

<div class="panel-body">

<div class="row">

<div class="col-sm-10">

<dt>

@Html.DisplayNameFor(model => model.Fam)

</dt>

<dd>

@Html.DisplayFor(model => model.Fam)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Ima)

</dt>

<dd>

@Html.DisplayFor(model => model.Ima)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Otc)

</dt>

<dd>

@Html.DisplayFor(model => model.Otc)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Dr)

</dt>

<dd>

@Html.DisplayFor(model => model.Dr)

</dd>

<hr />

<dt>

@Html.DisplayNameFor(model => model.Pol.Txt)

</dt>

<dd>

@Html.DisplayFor(model => model.Pol.Txt)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Nac.Txt)

</dt>

<dd>

@Html.DisplayFor(model => model.Nac.Txt)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Vneshnost)

</dt>

<dd>

@Html.DisplayFor(model => model.Vneshnost)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Pasport)

</dt>

<dd>

@Html.DisplayFor(model => model.Pasport)

</dd>

<hr />

<b><h4>Место рождения:</h4></b>

<dt>

@Html.DisplayNameFor(model => model.Obl.Txt)

</dt>

<dd>

@Html.DisplayFor(model => model.Obl.Txt)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Rn.Txt)

</dt>

<dd>

@Html.DisplayFor(model => model.Rn.Txt)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Np.Txt)

</dt>

<dd>

@Html.DisplayFor(model => model.Np.Txt)

</dd>

</div>

<div class="col-sm-2">

@if (Model.Foto != null)

{

@Html.Raw("<img class=\"img-responsive\" src=\"data:image/jpeg;base64," + Convert.ToBase64String(Model.Foto) + "\" />")

}

</div>

</div>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingTwo">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseTwo" aria-expanded="false" aria-controls="collapseTwo">

Адреса:

</a>

</h4>

</div>

<div id="collapseTwo" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTwo">

<div class="panel-body">

<ul>

@foreach (Address c in Model.Addresses)

{

@:

<b>Область: </b> @c.Obl.Txt

@:

<b>Район:</b> @c.Rn.Txt

@:

<b>Населенный пункт:</b> @c.Np.Txt

@:

<b>Улица:</b> @c.Ul

@:

<b>Дом:</b> @c.Dom

@:

<b>Корпус:</b> @c.Korpus

@:

<b>Квартира</b> @c.Kvartira

@:

<b>Примечание</b> @c.Prim

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingNine">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseNine" aria-expanded="false" aria-controls="collapseNine">

Образования:

</a>

</h4>

</div>

<div id="collapseNine" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingNine">

<div class="panel-body">

<ul>

@foreach (Educ c in Model.Educs)

{

<li> @c.Txt</li>

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingThree">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseThree" aria-expanded="false" aria-controls="collapseThree">

Профессии:

</a>

</h4>

</div>

<div id="collapseThree" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingThree">

<div class="panel-body">

<ul>

@foreach (Prof c in Model.Profs)

{

<li> @c.Txt</li>

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingFour">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseFour" aria-expanded="false" aria-controls="collapseFour">

Специальности:

</a>

</h4>

</div>

<div id="collapseFour" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingFour">

<div class="panel-body">

<ul>

@foreach (Spec c in Model.Specs)

{

<li> @c.Txt</li>

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingFive">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseFive" aria-expanded="false" aria-controls="collapseFive">

Осуждения:

</a>

</h4>

</div>

<div id="collapseFive" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingFive">

<div class="panel-body">

<ul>

@foreach (UK c in Model.UKs)

{

@:

<b>Пункт УК: </b> @c.Point

@:

<b>Часть УК: </b> @c.Part

@:

<b>Статья УК: </b> @c.Article

@:

<b>Название: </b> @c.Name

<br />

@:

<b>Фабула: </b> @c.Fabula

<br />

@:

<b>Примечания: </b> @c.Prim

<br />

@:

<b>Дата вынесения приговора: </b> @c.DatePrig.ToShortDateString()

<br />

@:

<b>Суд вынесший решение: </b> @c.Court.Name

@:

<b>Фамилия судьи: </b> @c.Fam

@:

<b>Имя судьи: </b> @c.Ima

@:

<b>Отчество: </b> @c.Otch

<br />

@:

<b>Местро отбывания наказания: </b> @c.Location

<br />

@:

<b>Имеется ли не снятая или непогашенная судимость?: </b> @c.Sudim

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingSix">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseSix" aria-expanded="false" aria-controls="collapseSix">

Правонарушения:

</a>

</h4>

</div>

<div id="collapseSix" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingSix">

<div class="panel-body">

<ul>

@foreach (Koap c in Model.Koaps)

{

@:

<b>Пункт КоАП: </b> @c.Point

@:

<b>Часть КоАП: </b> @c.Part

@:

<b>Статья КоАП: </b> @c.Article

@:

<b>Название: </b> @c.Name

<br />

@:

<b>Фабула: </b> @c.Fabula

<br />

@:

<b>Примечания: </b> @c.Prim

<br />

@:

<b>Дата решения: </b> @c.DateRe.ToShortDateString()

<br />

@:

<b>Суд вынесший решение: </b> @c.Court.Name

@:

<b>Фамилия судьи: </b> @c.Fam

@:

<b>Имя судьи: </b> @c.Ima

@:

<b>Отчество: </b> @c.Otch

<br />

@:

<b>Вид административного взыскания: </b> @c.Vidvz

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingSeven">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseSeven" aria-expanded="false" aria-controls="collapseSeven">

Медицинские комиссии:

</a>

</h4>

</div>

<div id="collapseSeven" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingSeven">

<div class="panel-body">

<ul>

@foreach (Medic c in Model.Medics)

{

@:

<b>Медицинская комиссия: </b> @c.MedcomId

@:

<b>Дата: </b> @c.Date.ToShortDateString()

@:

<b>Результат: </b> @c.Result

<br />

@:

<b>Фамилия врача: </b> @c.Fam

@:

<b>Имя врача: </b> @c.Ima

@:

<b>Отчество врача: </b> @c.Otch

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingTen">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseTen" aria-expanded="false" aria-controls="collapseTen">

Доставление/оформление лица:

</a>

</h4>

</div>

<div id="collapseTen" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTen">

<div class="panel-body">

<ul>

@foreach (Naprav c in Model.Napravs)

{

@:

<b>Основания направления: </b> @c.Osnnap.Txt

@:

<b>Дата помещения в ЛТП: </b> @c.DataPom.ToShortDateString()

<br />

<br />

@:

<b>Фамилия сотрудника, доставивший лицо: </b> @c.FamD

<br />

@:

<b>Имя сотрудника, доставившего лицо: </b> @c.ImaD

<br />

@:

<b>Отчество сотрудника, доставившего лицо: </b> @c.OtcD

<br />

@:

<b>Телефон сотрудника, доставившего лицо: </b> @c.TelD

<br />

<br />

@:

<b>Фамилия сотрудника, оформившего лицо: </b> @c.FamO

<br />

@:

<b>Имя сотрудника, оформившего лицо: </b> @c.ImaO

<br />

@:

<b>Отчество сотрудника, оформившего лицо: </b> @c.OtcO

<br />

@:

<b>Телефон сотрудника, оформившего лицо: </b> @c.TelO

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingEleven">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseEleven" aria-expanded="false" aria-controls="collapseEleven">

Обжалования решения о направлении в ЛТП:

</a>

</h4>

</div>

<div id="collapseEleven" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingEleven">

<div class="panel-body">

<ul>

@foreach (Objal c in Model.Objals)

{

@:

<b>Фамилия подавшего заявление: </b> @c.Fam

<br />

@:

<b>Имя подавшего заявление: </b> @c.Ima

<br />

@:

<b>Отчество подавшего заявление: </b> @c.Otch

<br />

@:

<b>Телефон подавшего заявление: </b> @c.Tel

@:

<b>Дата обжалования: </b> @c.Date.ToShortDateString()

@:

<b>Результат: </b> @c.Result

<br />

@:

<b>Во время нахождения в ЛТП: </b> @c.Locat

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingEight">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseEight" aria-expanded="false" aria-controls="collapseEight">

Превентивные надзоры:

</a>

</h4>

</div>

<div id="collapseEight" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingEight">

<div class="panel-body">

<ul>

@foreach (Privent c in Model.Privents)

{

@:

<b>Дата установления: </b> @c.Data.ToShortDateString()

@:

<b>Срок: </b> @c.Srok

<br />

@:

<b>Уголовно-исполнительная инспекция осуществляющая контроль: </b> @c.Inspec

<br />

@:

<b>Фамилия ответственого сотрудника: </b> @c.Fam

@:

<b>Имя ответственого сотрудника: </b> @c.Ima

@:

<b>Отчество ответственого сотрудника: </b> @c.Otc

<br />

@:

<b>Телефон ответственого сотрудника: </b> @c.Tel

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingTwelve">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseTwelve" aria-expanded="false" aria-controls="collapseTwelve">

Социально полезные связи:

</a>

</h4>

</div>

<div id="collapseTwelve" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTwelve">

<div class="panel-body">

<ul>

@foreach (Svaz c in Model.Svazs)

{

@:

<b>Фамилия: </b> @c.Fam

<br />

@:

<b>Имя: </b> @c.Ima

<br />

@:

<b>Отчество: </b> @c.Otc

<br />

@:

<b>Дата рождения: </b> @c.Dr.ToShortDateString()

<br />

@:

<b>Вид связи: </b> @c.Vidsv.Txt

<br />

@:

<b>Примечание: </b> @c.Prim

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingTvelve">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseTvelve" aria-expanded="false" aria-controls="collapseTvelve">

Отряды в ЛТП:

</a>

</h4>

</div>

<div id="collapseTvelve" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTvelve">

<div class="panel-body">

<ul>

@foreach (Otrad c in Model.Otrads)

{

@:

<b>ЛТП: </b> @c.LTP.Nom

@:

<b>Дата помещения: </b> @c.Date.ToShortDateString();

@:

<b>Номер отряда: </b> @c.NomOtr

<br />

@:

<b>Примечание: </b> @c.Prim

<br />

@:

<b>Должность начальника отряда: </b> @c.Dol

@:

<b>Звание начальника отряда: </b> @c.Zv

<br />

@:

<b>Фамилия начальника отряда: </b> @c.Fam

@:

<b>Имя начальника отряда: </b> @c.Ima

@:

<b>Отчество начальника отряда: </b> @c.Otch

<br />

@:

<b>Телефон начальника отряда: </b> @c.Tel

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingThirteen">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseThirteen" aria-expanded="false" aria-controls="collapseThirteen">

Задолженности:

</a>

</h4>

</div>

<div id="collapseThirteen" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingThirteen">

<div class="panel-body">

<ul>

@foreach (Debt c in Model.Debts)

{

@:

<b>Вид задолженности: </b> @c.Viddebt.Txt

@:

<b>Сумма: </b> @c.Sum

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingFriteen">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseFriteen" aria-expanded="false" aria-controls="collapseFriteen">

Трудовая деятельность до направления в ЛТП:

</a>

</h4>

</div>

<div id="collapseFriteen" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingFriteen">

<div class="panel-body">

<ul>

@foreach (Work c in Model.Works)

{

@:

<b>Место работы до направления в ЛТП: </b> @c.Job

<br />

@:

<b>Занимаемая должность: </b> @c.Job

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingFourteen">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseFourteen" aria-expanded="false" aria-controls="collapseFourteen">

Трудовая деятельность в ЛТП:

</a>

</h4>

</div>

<div id="collapseFourteen" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingFourteen">

<div class="panel-body">

<ul>

@foreach (WorkLTP c in Model.WorkLTPs)

{

@:

<b>Отношение к труду: </b> @c.Otn

<br />

@:

<b>Трудоустройство в ЛТП: </b> @c.Trud

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingFifteen">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseFifteen" aria-expanded="false" aria-controls="collapseFifteen">

Зависимости:

</a>

</h4>

</div>

<div id="collapseFifteen" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingFifteen">

<div class="panel-body">

<ul>

@foreach (Zavis c in Model.Zaviss)

{

@:

<b>Зависимость: </b> @c.Zav

<br />

@:

<b>Принятая мера: </b> @c.Prin

<br />

@:

<b>Дата: </b> @c.Date.ToShortDateString()

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingSixteen">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseSixteen" aria-expanded="false" aria-controls="collapseSixteen">

Информация врача-нарколога:

</a>

</h4>

</div>

<div id="collapseSixteen" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingSixteen">

<div class="panel-body">

<ul>

@foreach (Lechen c in Model.Lechens)

{

@:

<b>Состоит у врача нарколога: </b> @c.Sost

<br />

@:

<b>Принятая мера: </b> @c.Prin

<br />

@:

<b>Дата: </b> @c.Date.ToShortDateString()

<br />

<hr />

}

</ul>

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingSeventeen">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseSeventeen" aria-expanded="false" aria-controls="collapseSeventeen">

Участие в общественной жизни ЛТП:

</a>

</h4>

</div>

<div id="collapseSeventeen" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingSeventeen">

<div class="panel-body">

<ul>

@foreach (Obchest c in Model.Obchests)

{

@:

<b>Участие: </b> @c.Uch

<br />

<hr />

}

</ul>

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingEighteen">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseEighteen" aria-expanded="false" aria-controls="collapseEighteen">

Отношение к лицам:

</a>

</h4>

</div>

<div id="collapseEighteen" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingEighteen">

<div class="panel-body">

<ul>

@foreach (Soderotnosh c in Model.Soderotnoshes)

{

@:

<b>ФИО: </b> @c.Name

<br />

@:

<b>Отношение: </b> @c.Otnosh.Txt

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingNineteen">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseNineteen" aria-expanded="false" aria-controls="collapseNineteen">

Мнение психолога:

</a>

</h4>

</div>

<div id="collapseNineteen" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingNineteen">

<div class="panel-body">

<ul>

@foreach (Psix c in Model.Psixes)

{

@:

<b>Информация психолога: </b> @c.Psi

<br />

@:

<b>Ф.И.О. психолога: </b> @c.Name

<br />

@:

<b>Дата: </b> @c.Date.ToShortDateString()

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingTwenty">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseTwenty" aria-expanded="false" aria-controls="collapseTwenty">

Дисциплинарные проступки:

</a>

</h4>

</div>

<div id="collapseTwenty" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTwenty">

<div class="panel-body">

<ul>

@foreach (Discip c in Model.Discips)

{

@:

<b>Дисциплинарный проступок: </b> @c.Dis

<br />

@:

<b>Дата совершения: </b> @c.Date.ToShortDateString()

<br />

@:

<b>Наказание: </b> @c.Nak

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingTwentyOne">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseTwentyOne" aria-expanded="false" aria-controls="collapseTwentyOne">

Самовольное оставление ЛТП:

</a>

</h4>

</div>

<div id="collapseTwentyOne" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTwentyOne">

<div class="panel-body">

<ul>

@foreach (Samohod c in Model.Samohods)

{

@:

<b>Дата побега: </b> @c.DateP.ToShortDateString()

@:

<b>Дата задержания: </b> @c.DateZ.ToShortDateString()

<br />

@:

<b>Ф.И.О. задержавшего сотрудника: </b> @c.FIO

<br />

@:

<b>Информация о побеге: </b> @c.Samovol.Txt

<br />

@:

<b>Примечание: </b> @c.Prim

<br />

<hr />

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingTwentyTwo">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseTwentyTwo" aria-expanded="false" aria-controls="collapseTwentyTwo">

Документы переданные при доставлении:

</a>

</h4>

</div>

<div id="collapseTwentyTwo" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTwentyTwo">

<div class="panel-body">

<ul>

@foreach (Doc c in Model.Docs)

{

<li>@c.Txt</li>

}

</ul>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading" role="tab" id="headingTwentyFri">

<h4 class="panel-title">

<a class="collapsed" role="button" data-toggle="collapse" data-parent="#accordion" href="#collapseTwentyFri" aria-expanded="false" aria-controls="collapseTwentyFri">

Информация о лице после освобождения из ЛТП:

</a>

</h4>

</div>

<div id="collapseTwentyFri" class="panel-collapse collapse" role="tabpanel" aria-labelledby="headingTwentyFri">

<div class="panel-body">

<ul>

@foreach (Osv c in Model.Osvs)

{

@:

<b>Дата освобождения: </b> @c.DateOsv.ToShortDateString()

@:

<b>Дата прибытия: </b> @c.DatePrib.ToShortDateString()

<br />

@:

<b>Cотрудник контролировавший прибытие: </b> @c.Control

<br />

@:

<b>Имеется ли место жительства: </b> @c.Locat

<br />

@:

<b>Необходимость доставки к дому: </b> @c.Dost

<br />

@:

<b>Информация о трудоустройстве после освобождения: </b> @c.Trud

<br />

@:

<b>Личная позиция лица о трудоустройсте после освобождения: </b> @c.Posic

<br />

<hr />

}

</ul>

</div>

</div>

</div>

</div>

</dl>

</div>

<p>

@Html.ActionLink("Назад", "Index")

</p>

Notify/Index.cshtml

@model IEnumerable<AISLTP.Entities.Notify>

@{

ViewBag.Title = "Уведомления";

}

<h2>@ViewBag.Title</h2>

<table class="table table-striped">

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@item.Lico.Fam

@item.Lico.Ima

@item.Lico.Otc

:

@item.Content

@Html.DisplayFor(model => item.Date)

</td>

</tr>

}

</tbody>

</table>

Messages/Index.cshtml

@model IEnumerable<AISLTP.Entities.Message>

@{

ViewBag.Title = "Сообщения пользователей";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>@ViewBag.Title</h2>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.Date)

</th>

<th>

@Html.DisplayNameFor(model => model.User.Login)

</th>

<th>

@Html.DisplayNameFor(model => model.Content)

</th>

<th>

@Html.DisplayNameFor(model => model.IsRead)

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Date)

</td>

<td>

@Html.DisplayFor(modelItem => item.User.Login)

</td>

<td>

@Html.DisplayFor(modelItem => item.Content)

</td>

<td>

@if (item.IsRead)

{ <span>Да</span> }

else

{ <span>Нет</span> }

</td>

<td>

@Html.ActionLink("Просмотреть", "Details", new { id=item.ID }, new { @class = "btn btn-default btn-sm", title = "Просмотреть" })

</td>

</tr>

}

</table>