|  |  |
| --- | --- |
|  |  |
| **Утверждаю**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ФИО«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2020  | **Утверждаю**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ФИО«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2020 |

**Legal Desk**

**Руководство по развертыванию продукта**

Действует с «16» сентября 2020 г.

На 17 листах

г. Москва, 2020

Содержание

[Содержание 2](#_Toc52462448)

[1. Сокращения, термины и определения 3](#_Toc52462449)

[2. Общие сведения 4](#_Toc52462450)

[3. Требования к оборудованию 5](#_Toc52462451)

[4. Общая информация 6](#_Toc52462452)

[5. Дизайн. Архитектура приложений 7](#_Toc52462453)

[5.1. Технология архитектуры 7](#_Toc52462454)

[5.1.1. Back-end 8](#_Toc52462455)

[5.1.2. Front-end 9](#_Toc52462456)

[5.1.3. База данных 9](#_Toc52462457)

[5.1.4. Внешние API 10](#_Toc52462458)

[6. Взаимодействие портала с внешними сервисами 11](#_Toc52462459)

[7. Система безопасности и требования контроля 13](#_Toc52462460)

[8. Конфигурация системы 14](#_Toc52462461)

[9. Риски и ограничения проекта 15](#_Toc52462462)

[Список рисунков 16](#_Toc52462463)

[Список таблиц 17](#_Toc52462464)

1. Сокращения, термины и определения

| Термин/Сокращение | Определение |
| --- | --- |
| SPA | Single Page Application – одностраничное веб-приложение. |
| БД | База данных. |
| Клиентский портал Legal Desk | Веб-сервис, который предназначен для:* работы с типовыми документами;
* получения юридических услуг и консультаций;
* отслеживания статуса текущих дел пользователя.
 |
| ПО | Программное обеспечение. |
| AD | Active Directory |

1. Общие сведения

В руководстве приведены общие сведения по архитектуре клиентского портала Legal Desk, всем составляющим портала, их реализации и взаимодействии друг с другом.

1. Требования к оборудованию

Для корректной работы программного продукта необходимо соблюдать следующие требования к оборудованию:

1. CPU: Intel® Core™ i7-6700.
2. RAM: 32 ГБ DDR4.
3. ROM: 120 ГБ SATA SSD.
4. Общая информация

Используемые инструменты:

* Используемые редакторы и IDE: Visual Studio Code, JetBrains WebStorm;
* Язык программирования – JavaScript;
* Node.js – программная среда на основе движка V8, для трансляции JavaScript кода в машинный код;
* NPM – пакетный менеджер модулей для node.js;
* MongoDB – документоориентированная система управления базами данных с открытым исходным кодом;
* Mongoose.js – библиотека для моделирования объектов MongoDB, разработанный для работы в асинхронной среде;
* Express.js – фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений;
* React.js – библиотека с открытым исходным кодом;
* Web-server Nginx 1.10.3 для реализации взаимодействия интерфейса предлагаемого ПО с бизнес-логикой по протоколам HTTP (HTTPS).
1. Дизайн. Архитектура приложений

Клиентский портал Legal Desk – веб-приложение, которое представляет собой SPA и позволяет работать с делами пользователей.

В системе предусмотрено две роли пользователя: Обычный пользователь и Администратор. Для администратора реализован отдельный интерфейс, где он может получить доступ к настройкам продуктов, системным настройкам и настройкам пользователей.



Рисунок 1. Схема ролей пользователей

* 1. Технология архитектуры

Обобщенная схема технической архитектуры Системы представлена на Рисунке 2.



Рисунок 2. Схема технологии архитектуры

* + 1. Back-end

Back-end часть реализована на языке программирования JavaScript с использованием программной платформы Node.js с открытым исходным кодом, основанной на движке V8, который транслирует JavaScript в машинный код.

Кроме встроенных пакетов Node.js посредством пакетного менеджера npm используются дополнительные утилиты:

* mongoosejs как конструктор объектов посредством схем для базы данных;
* expressjs как фреймворк для конструирования сервера;
* passportjs для реализации функций авторизации и др.

Код написан с применением паттерна MVC для разделения логики, представления и данных и встроенной в Node.js кластеризации для распределения нагрузки и поддержания отказоустойчивости приложения.

В качестве модели используются mongoosejs schema для формирования объектов для базы данных.

В качестве представления используются статичные шаблоны, генерируемые на стороне сервера с помощью expressjs template's.

Контроллеры представляют логику приложения, включая логику общения со всеми компонентами приложения.

* + 1. Front-end

Front-end состоит из ejs представлений с подключаемым JavaScript кодом, которые генерируются на стороне сервера. Код разделен на контроллеры, каждый из которых отвечает за свою часть логики. Связь front-end и back-end части выполняется по протоколам REST посредством проксирования HTTP/HTTPS запросов. Front-end разделен на две составляющих:

* личный кабинет пользователя – для работы с делами пользователя;
* раздел администратора – для настройки подключения к компонентам, настройки ролей пользователей и их личной информации, для настройки продуктов.
	+ 1. База данных

В качестве хранилища для данных используется MongoDB – документоориентированная система управления базами данных с открытым исходным кодом. Доступ к базе осуществляется только со стороны back-end.

Для хранения данных используется главная база и базы для каждого из развернутых инстансов.

Таблица 1. Описание структуры баз данных

| БД | Collections | Description |
| --- | --- | --- |
| 2ndMain | instances | коллекция инстансов |
| sessions | коллекция сессий всех инстансов |
| users | коллекция пользователей базы |
| 2nd\_nameinstance\_2nd\_law (nameinstance – название инстанса) | cases | коллекция дел |
| categories | коллекция категорий пользователя |
| dictionaries | коллекция справочников системы |
| files | коллекция, содержащая информацию о файлах |
|  | folders | коллекция, содержащая информацию о папках |
| products | коллекция продуктов |
| status | коллекция, содержащая информацию о статусах |
| tasks | коллекция, содержащая информацию о задачах |
| users | коллекция пользователей инстанса |

* + 1. Внешние API

Для реализации бизнес логики ПО используются следующие компоненты:

* cases service – для хранения информации по делам пользователей;
* files service – для хранения файлов;
* conversations service – для подключения чат-ботов;
* documents service – для генерации документов на основе шаблонов.

Общение с компонентами выполняется по протоколам REST посредством HTTP/HTTPS запросов. Часть данных для сохранения независимости портала хранится в базе MongoDB.

Также в реализации связи с компонентами есть скрипты, написанные на языке программирования JavaScript. Скрипты предназначены для запуска по расписанию посредством встроенной в Linux утилиты cron, для ежедневной синхронизации данных по всем компонентам.

1. Взаимодействие портала с внешними сервисами

Портал взаимодействует со следующими сервисами:

1. Caseone в части ведения дела. Если при создании продукта было указано, что необходимо создавать дело, то после работы пользователя с продуктом дело также будет автоматически создано в Caseone.

В качестве ответственного по делу в Caseone будет установлен пользователь (указывается в настройках портала), от имени которого создается дело.

1. Form.One в части работы с ботами, например, при оформлении договоров (подробное описание приведено в руководстве пользователя клиентского портала 2Law);
2. Doc.One в части работы с документами, шаблонами документов (подробное описание приведено в руководстве пользователя клиентского портала 2Law);
3. Active Directory (AD) для авторизации на портале. Полная синхронизация всех пользователей с AD происходит каждый раз при запуске портала, а также выполняется периодически согласно настройке в web.config (по умолчанию ежедневно в полночь). Первичная привязка пользователя портала к пользователю AD осуществляется по совпадению электронной почты пользователей, после этого у пользователя портала проставляется objectGUID пользователя из AD и дальнейшая синхронизация осуществляется по этому идентификатору.

Доступность информации пользователю во внешних сервисах зависит от настроек продукта при его создании.

Взаимодействие портала с внешними сервисами выполняется с помощью внутренней интеграции компонентов систем и следующих вебхуков:

1. /api/v1.0/webhook/updateCase (deprecated) – предназначен для обновления следующей информации по делу: название дела, стадия дела, статус дела: «В архиве», «В работе», «Открыт в 2ndLaw»:
* method: POST;
* header: «Content-Type: application/json»;
* body: {dialogId: ""} или {caseId: ""}.
1. /api/v1.0/getCaseIdByDialog – предназначен для получения идентификатора дела при наличии только идентификатора диалога:
* method: POST;
* header: «Content-Type: application/json»;
* body: {dialogId: ""}.

|  |
| --- |
|  |
| Внимание! Метод является устаревшим и будет удален в последующих релизах. |
|  |

1. /api/v1.0/webhook/updateCaseOne – предназначен для синхронизации дел между порталом и Caseone. Со стороны Caseone приходит общая информация по делу, включая: название дела, стадию дела, статуса: «В архиве», «В работе», «Открыт в 2ndLaw», со стороны портала выполняется трансфер данных из Form.one в Caseone.
* method: POST;
* header: «Content-Type: application/json»;
* body: {dialogId: ""} или {caseId: ""}.
1. /api/v1.0/webhook/updateTasks – предназначен для синхронизации задач делу из Caseone в портале по идентификатору дела – caseId или идентификатору диалога – dialogId:
* method: POST;
* header: «Content-Type: application/json»;
* body: {dialogId: ""} или {caseId: ""}.
1. /api/v1.0/webhook/updateFolders – предназначен для синхронизации папок из Caseone в портале по идентификатору дела – caseId или идентификатору диалога – dialogId:
* method: POST;
* header: «Content-Type: application/json»;
* body: {dialogId: ""} или {caseId: ""}.
1. /api/v1.0/createCOcaseManual – предназначен для ручного создания дела в Caseone. Вебхук используется, если в настройке продукта установлено значение «Нет» для поля «Создать дело»:
* method: POST;
* header: «Content-Type: application/json»;
* body: {dialogId: ""} или {caseId: ""}.
1. https://my.2nd.law/api/v1.0/cases/getUserByDialogOrCaseId – предназначен для получения информации о пользователе портала:
* method: POST;
* header: «Content-Type: application/json»;
* body: { “dialogId”: “@cuuid” }.

Настройка параметров для взаимодействия с AD выполняется в web.config:

* domainName – имя домена, в котором расположены синхронизируемые пользователи (обязательный параметр);
* netBiosDomainName – имя домена, указываемое пользователями при входе в систему (необязательный параметр; если не указан, используется domainName);
* container – выделенное имя (distinguished name) корневого контейнера, который содержит синхронизируемые группы (обязательный параметр);
* userName – имя пользователя для подключения к службе AD (необязательный, если сервер расположен в домене);
* password – пароль пользователя для подключения к службе AD (необязательный, если сервер расположен в домене);
* mappings – список синхронизируемых групп AD;
* groupName – имя группы в AD;
* roles – список ролей portal, сопоставленных с группой в AD;
* listeners – список OU, которые необходимо прослушивать на предмет изменений, чтобы синхронизировать пользователей с AD в реальном времени;
* distinguishedName – выделенное имя прослушиваемой OU.
1. Система безопасности и требования контроля

Используется SSL протокол типа PKCS #1 SHA-256 с шифрованием RSA.

Ограничение количества одновременных запросов к серверу с помощью модуля express-rate-limit.

Встроенная защита mongoose от атак путем инъекций вредоносного кода.

Ограничение количества попыток авторизации с помощью модуля express-brute.

1. Конфигурация системы

Из учета используемых компонентов и утилит, для нормальной работы приложения необходима следующая конфигурация оборудования сервера (см. Таблица 2).

Таблица 2. Описание конфигурации сервера

| CPU | RAM | Persistent storage |
| --- | --- | --- |
| 16 cores, 3500-6700 | 32 GB DDR4 | 2x 512 GB SATA SSD |

1. Риски и ограничения проекта

В связи с отсутствием данных о количестве пользователей, которые будут использовать ПО, единственным ограничением можно считать конфигурацию системы, которая может оказаться слишком слабой для конечного количества пользователей, если это количество будет достаточно большое.

В текущей конфигурации (Intel® Core™ i7-6700 Quad-Core Skylake с технологией Hyper-Threading, RAM 32 ГБ DDR4, 120 ГБ SATA SSD, видеокарта GeForce® GTX 1080) сервер способен обеспечить производительность для работы свыше 2000 пользователе й одновременно.

Список рисунков

[Рисунок 1. Схема ролей пользователей 7](#_Toc52462465)

[Рисунок 2. Схема технологии архитектуры 8](#_Toc52462466)

 Список таблиц

[Таблица 1. Описание структуры баз данных 9](#_Toc52462467)

[Таблица 2. Описание конфигурации сервера 14](#_Toc52462468)