МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
к.т.н., доцент кафедры САУ

_____ Н.В. Плотникова

“___”_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2019 г.

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ КОНТРОЛЯ СОТРУДНИКОВ ЗА РАБОТОЙ НА КОМПЬЮТЕРЕ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2019.115-111.ВКР

Научный руководитель,
старший преподаватель кафедры
системного программирования
_____ К.В. Репина

Автор работы,
студент группы КЭ-401
_____ В.А. Иоговский

Ученый секретарь
(нормоконтролер)
_____ О.Н. Иванова
“___”_____ 2019 г.

Челябинск-2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
“___”_____ 2019 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-401
Иоговскому Всеволоду Алексеевичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 25.04.2019 № 899)

   Разработка клиент-серверного приложения для контроля сотрудников за работой на компьютере

2. **Срок сдачи студентом законченной работы:** 05.06.2019.

3. **Исходные данные к работе**

   3.1. Головатый А., Каплан-Мосс Дж. Django. Подробное руководство, 2-е изд. Пер. с англ.–СПб.:Символ-Плюс, 2013.–560 с.
   3.2. Документация по Python [Электронный ресурс] URL: https://docs.python.org (дата обращения 29.04.2019)

4. **Перечень подлежащих разработке вопросов**

4.1. Выполнить анализ предметной области
4.2. Спроектировать клиент-серверное приложение
4.3. Реализовать и протестировать клиент-серверное приложение

5. **Дата выдачи задания:** 08.02.2019.

Научный руководитель

Старший преподаватель кафедры СП К.В. Репина

Задание принял к исполнению В.А. Иоговский

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

Каждое предприятие стремится максимизировать свою эффективность, обращаясь за помощью к специалистам, устанавливая современное оборудование и разрабатывая новые методы ведения бизнеса. Но старания работодателя могут оказаться бессмысленными при наличии безответственных сотрудников, которые впустую тратят рабочий ресурс предприятия. Поэтому контроль за деятельностью работника необходим для обеспечения надлежащего функционирования предприятия.

Существует множество видов деятельности, на которые сотрудники тратят рабочее время. Например, общение с коллегами, избыточные встречи и бесполезная деятельность за рабочими компьютерами, о последнем и пойдет речь далее. По данным 2012 года, в Российской Федерации почти 50 % всех работников используют компьютер на рабочем месте [4], а по результатам опроса американского сервиса для подбора персонала, из числа сотрудников в возрасте от 18 до 35 лет – 73 % респондентов сообщили, что ежедневно проводят рабочее время ненадлежащим образом, а 64 %, что каждый день посещают сайты, не связанные с работой [2]. Поэтому эффективность должна поддерживаться в том числе и за рабочими компьютерами, с помощью использования комплексов, созданными для наблюдения и контроля за работой сотрудников за компьютерами.

Самый близкий по смыслу термин для описания подобных программных комплексов – это DLP-система. DLP-система (Data Leak Prevention) – специальный программный комплекс, предотвращающий утечки данных. Кроме своей основной функции, DLP-системы так же хорошо подходят для контроля использования рабочего времени и рабочих ресурсов сотрудниками [18]. По опросу сайта SearchInform.ru от 2018 года, только 32 % компаний используют DLP-систему на своем предприятии [12].

Большая часть этих программных комплексов представляет собой клиент-серверные приложения, в которых клиент устанавливается на компьютер работника, отслеживает его действия, общую активность и отправляет на сервер. Серверная же часть позволяет хранить и систематизировать полученную информацию, а администратору наблюдать за происходящим у сотрудников на рабочих компьютерах.

Цели и задачи

Главной целью данной работы является разработка клиента и сервера для системы контроля сотрудников за работой на компьютере. Клиент, установленный на компьютер работника должен быть настраиваемым, скрытно работать все время, что компьютер запущен, собирать и отправлять данные на сервер. Сервер же, в первую очередь, должен предоставлять возможность администратору наблюдать за информацией, полученной от конкретной запущенной копии клиента, а также хранить, систематизировать и удалять старые записи, чтобы избежать большого количества занятого пространства.

Для достижения поставленной цели необходимо решить следующие задачи:

1) проанализировать требования к системе;
2) спроектировать архитектуру клиента и сервера;
3) реализовать клиент и сервер;
4) протестировать систему.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения и списка использованной литературы. Объем работы составляет 46 страниц, список литературы содержит 19 источников.

В первой главе устанавливается задача, обсуждается правовой аспект применение систем контроля сотрудников, обозреваются и сравниваются аналоги создаваемой системы.

Во второй главе определяются функциональные и нефункциональные требования к системе, описываются варианты ее использования. Проводится проектирование архитектуры системы, а также проектирование базы данных.

В третьей главе проводится выбор средств разработки клиента и сервера, реализация серверной части, включая структуру базы данных, внутреннюю логику и интерфейс администратора, и реализацию клиента.

В четвертой главе приводятся результаты функционального тестирования разработанной системы.

В заключении сделаны выводы о проделанной работе.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Постановка задачи

Внедрение систем контроля за работой сотрудников на компьютерах необходимо для решения следующего списка задач:

- общее отслеживание деятельности сотрудников за рабочими компьютерами;
- выявление неэффективных сотрудников;
- выявление сотрудников, которые могут допустить или уже допустили утечку конфиденциальных данных, коммерческой тайны или нарушили соглашение о неразглашении;
- отслеживание времени, проведенного сотрудниками не по направлению их рабочей деятельности;
- выявление запрещенного программного обеспечения, запущенного на компьютерах сотрудников;
- использование превентивных мер в отношении сотрудников.

Для предприятий внедрение систем контроля сотрудников за работой на компьютерах является решением, способным повысить эффективность и безопасность всего бизнеса, сэкономить средства, которые могли быть потрачены на оплату труда непродуктивных сотрудников.

Для обеспечения правильной работы такой системы, она должна включать в себя локальный клиент, устанавливаемый на компьютер сотрудников и серверную часть с административным сервисом. В рамках данной работы они будут реализованы.

1.2. Юридический аспект применения

При применении систем контроля сотрудников за работой на компьютере возникает вопрос о законности применения подобных систем.

Согласно статье 23 Конституции Российской Федерации (п. 1) каждый имеет право на неприкосновенность частной жизни, личную и семейную тайну, защиту своей чести и доброго имени.

Конституцией Российской Федерации также установлено: каждый имеет право на тайну переписки, телефонных переговоров, почтовых, телеграфных и иных сообщений. Ограничение этого права допускается только на основании судебного решения (п. 2 ст. 23).

Статьей 24 Конституции Российской Федерации (п. 1) определено, что сбор, хранение, использование и распространение информации о частной жизни лица без его согласия не допускаются.

При этом пунктом 4 статьи 29 Конституции Российской Федерации установлено, что каждый имеет право свободно искать, получать, передавать, производить и распространять информацию любым законным способом.

Гражданским кодексом Российской Федерации установлен порядок охраны частной жизни гражданина (статья 152.2, п. 1), в частности, указано, что если иное прямо не предусмотрено законом, не допускаются без согласия гражданина сбор, хранение, распространение и использование любой информации о его частной жизни, в частности сведений о его происхождении, о месте его пребывания или жительства, о личной и семейной жизни. Между тем, не являются нарушением этих правил сбор, хранение, распространение и использование информации о частной жизни гражданина в государственных, общественных или иных публичных интересах, а также в случаях, если информация о частной жизни гражданина ранее стала общедоступной либо была раскрыта самим гражданином или по его воле.

Гражданским кодексом Российской Федерации также установлено (статья 152.2, п. 2), что стороны обязательства не вправе разглашать ставшую известной им при возникновении и (или) исполнении обязательства

информацию о частной жизни гражданина, являющегося стороной или третьим лицом в данном обязательстве, если соглашением не предусмотрена возможность такого разглашения информации о сторонах.

Кодекс также определяет, что в случаях, когда информация о частной жизни гражданина, полученная с нарушением закона, содержится в документах, видеозаписях или на иных материальных носителях, гражданин вправе обратиться в суд с требованием об удалении соответствующей информации, а также о пресечении или запрещении дальнейшего ее распространения путем изъятия и уничтожения без какой бы то ни было компенсации изготовленных в целях введения в гражданский оборот экземпляров материальных носителей, содержащих соответствующую информацию, если без уничтожения таких экземпляров материальных носителей удаление соответствующей информации невозможно (п. 4 ст. 152.2 ГК РФ).

Федеральный закон от 27 июля 2006 года № 152-ФЗ «О персональных данных» (далее – Закон) более широко определяет объект защиты: под персональными данными, согласно статье 3 Закона понимается любая информация, относящаяся прямо или косвенно к определенному или определяемому физическому лицу (субъекту персональных данных), обработку которой осуществляет оператор – государственный орган, муниципальный орган, юридическое или физическое лицо, самостоятельно или совместно с другими лицами организующие и (или) осуществляющие обработку персональных данных, а также определяющие цели обработки персональных данных, состав персональных данных, подлежащих обработке, действия (операции), совершаемые с персональными данными.

Обработка персональных данных допускается в случаях, установленных Законом, в частности, если такая обработка персональных данных осуществляется с согласия субъекта персональных данных (пп. 1 п. 1 статьи 6 Закона).

Статьей 86 Трудового кодекса Российской Федерации устанавливается, что обработка персональных данных работника может осуществляться

исключительно в целях обеспечения соблюдения законов и иных нормативных правовых актов, содействия работникам в трудоустройстве, получении образования и продвижении по службе, обеспечения личной безопасности работников, контроля количества и качества выполняемой работы и обеспечения сохранности имущества (п. 1).

Пунктом 1 статьи 9 Закона предусмотрено, что субъект персональных данных принимает решение о предоставлении его персональных данных и дает согласие на их обработку свободно, своей волей и в своем интересе. Согласие на обработку персональных данных должно быть конкретным, информированным и сознательным. Согласие на обработку персональных данных может быть дано субъектом персональных данных или его представителем в любой позволяющей подтвердить факт его получения форме.

Согласие на обработку персональных данных может быть отозвано субъектом персональных данных (п. 2 статьи 9 Закона). В случае отзыва субъектом персональных данных согласия на обработку персональных данных оператор вправе продолжить обработку персональных данных без согласия субъекта персональных данных при наличии оснований, указанных в Законе, в том числе, в Законе указано такое основание, как обработка персональных данных в целях исполнения договора, стороной которого является субъект персональных данных, в том числе в случае реализации оператором своего права на уступку прав (требований) по такому договору (пп. 5 п. 1 статьи 6 Закона).

На основании статьи 22 Закона требуется уведомление оператором до начала обработки персональных данных уполномоченного органа по защите прав субъектов персональных данных о своем намерении осуществлять обработку персональных данных, за исключением случаев, когда персональные данные обрабатываются в соответствии с трудовым законодательством, а также если персональные данные получены оператором в связи с заключением договора, стороной которого является субъект персональных дан-

ных, если персональные данные не распространяются, а также не предоставляются третьим лицам без согласия субъекта персональных данных и используются оператором исключительно для исполнения указанного договора и заключения договоров с субъектом персональных данных [19].

Таким образом, можно сделать следующие выводы [19]:

1. сбор и обработка информации, относящаяся прямо или косвенно к определенному физическому лицу, допускается на законных основаниях;
2. сбор и обработка информации может осуществляться оператором как самостоятельно, так и совместно с другими лицами;
3. обработка информации о работнике может осуществляться в целях контроля количества и качества выполняемой работы и обеспечения сохранности имущества;
4. законным основанием сбора и обработки персональных данных физического лица является его свободное, в своей воле и в своем интересе данное письменное согласие;
5. согласие на обработку персональных данных может быть отозвано, однако обработка персональных данных может быть продолжена без согласия лица в целях исполнения договора;
6. уведомление уполномоченного органа о намерении осуществлять обработку персональных данных работника не требуется.

Законность применения систем для контроля сотрудников за работой на компьютере также подтверждает и то, что существуют аналоги системы, используемые, в том числе, и государственными компаниями [15].

1.3. Обзор существующих аналогов

В связи с наличием проблемы с эффективностью сотрудников при работе на компьютерах, у работодателей есть широкий выбор программных комплексов, отслеживающих действия работника на рабочем месте. Рассмотрим плюсы, минусы и некоторые особенности нескольких из них.

1.3.1 NeoSpy

NeoSpy – программа для слежения за компьютером пользователей. Ее можно использовать в качестве клавиатурного шпиона, комплекса для родительского контроля и других видов онлайн слежения [16].

Плюсами NeoSpy можно назвать бесплатные консультации по работе с программой, присутствие клавиатурного шпиона (кейлоггера) и автоматическую установку с заранее заданными параметрами. Минусами же являются такие вещи как: отсутствие бесплатных версий и техническая поддержка только в самой дорогостоящей версии. Стоимость версии без ограничений – 2000 рублей.

1.3.2. Стахановец

Стахановец – программный комплекс, предназначенный для комплексного качественного повышения эффективности бизнеса, защиты информации и мониторинга персонала. С помощью комплекса Стахановец можно выявить кто из сотрудников крадет конфиденциальную информацию, работает на конкурентов, предлагает или берет откаты, собирается уволиться, тратит время на социальные сети [15].

Плюсами данного клиент-серверного приложения является: постоянная запись видео с экрана и аудио с микрофона/динамиков с сохранением в зашифрованном виде на машинах сотрудников для дальнейшего детального анализа в случае расследования инцидентов, анализатор рисков и производительности, перехват голосовых переговоров, видео с веб-камеры, POP3/SMTP трафика, геолокации и другое. Главным и, по существу, единственным минусом приложения является его стоимость за версию без ограничений, которая составляет до 9880 рублей за одну наблюдаемую клиентскую машину.

1.3.3. Crocotime

Crocotime – автоматическая программная система, служащая, главным образом, для того, чтобы собирать данные об использовании сотрудни-

ком рабочего времени. Клиенты начинают мониторинг с момента включения компьютера, что автоматически заносится в отчет как начало рабочего дня. Crocotime показывает количество отработанного персоналом времени за ПК в конкретной программе или на конкретном сайте [8].

К плюсам этого комплекса можно отнести бесплатный пробный период, отличный анализ статистики по времени, проведенному за конкретной задачей, а также его цену. Имеются и минусы, такие как: отсутствие возможностей слежения за конкретными действиями сотрудников, в том числе скриншотов, видеозаписей и так далее. Именно из-за этого данная программная система подойдет не всем. Цена облачной версии – 250 рублей за один отслеживаемый компьютер, либо по запросу для версии с сервером в локальной сети.

1.3.4. KickIdler

KickIdler – комплекс онлайн наблюдения за сотрудниками, с помощью которого можно проводить онлайн мониторинг рабочих компьютеров, записывать видео с экранов, использовать клавиатурного шпиона, осуществлять удаленное управление компьютером работника, а также проводить анализ продуктивности и составлять отчеты по рабочему времени [14].

К плюсам относится широкий функционал, кроссплатформенность, скрытый режим работы клиента и бесплатная пожизненная версия с ограничениями для шести компьютеров. Минусом является отсутствие веб-версии сервера и необходимость его установки вручную, а также малое количество примеров работы на сайте системы. Цена – от 600 рублей в месяц за один отслеживаемый компьютер до 10000 рублей за версию без ограничений по времени для одного ПК.

1.4. Сравнение существующих аналогов

В таблице 1 проведено сравнение описанных выше аналогов по одинаковым параметрам.

Табл. 1. Сравнение существующих аналогов

Параметр / Система	Снятие скриншотов	Администрирование	Платформа	Лицензия на 1 ПК
NeoSpy	+	Программа и Web (платно)		От 840 до 2000 рублей
Стахановец	+	Web (локальный или в интернете)		От 2200 ₽/год до 9800 руб.
CrocoTime	-	Программа и Web		250 ₽/мес.
KickIdler	+	Программа		От 600 ₽/мес. до 10000 руб.

Вывод

В первой главе был поставлен список задач, решаемых клиент-серверным приложением для контроля сотрудников за работой на компьютере, выделены правовые аспекты применения подобных систем, а также описаны существующие аналоги разрабатываемой системы, определены их положительные и отрицательные составляющие и проведено их сравнение.

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1. Определение требований к системе

Принцип работы клиент-серверного приложения для контроля сотрудников за работой на компьютере заключается в следующем. Существует локальный клиент, который при первом запуске записывается на компьютер работника и в последствии запускается вместе с компьютером. Клиент, используя предварительно заданные настройки, отправляет на сервер скриншот экрана работника и список запущенных процессов, помечая подозрительные. Сервер же принимает, сохраняет и обрабатывает данные, с помощью веб-интерфейса позволяет администратору просматривать информацию о компьютерах сотрудников и манипулировать ей.

2.1.1. Функциональные требования

Сервер системы, он же сервис администратора, должен обеспечивать добавление, хранение, систематизацию, автоматического удаление, редактирование и просмотр данных сотрудников администратором.

Локальный клиент должен устанавливаться на компьютер сотрудника, быть настраиваемым, запускаться со стартом компьютера, создавать и отправлять скриншоты и проанализированный список процессов пользователя.

Функциональные требования определяют функциональность программного обеспечения, то есть описывают, какие возможности должна предоставлять разрабатываемая система. Функциональные требования включают в себя бизнес-требования и пользовательские требования.

Были выявлены следующие функциональные требования для клиент-серверного приложения для контроля сотрудников за работой на компьютере:

1) сервер должен позволять принимать данные, посылаемые локальным клиентом;

2) сервер должен позволять хранить данные, полученные от локального клиента;

3) сервер должен обрабатывать и систематизировать данные, полученные от локального клиента;

4) сервер должен автоматически удалять старые данные, чтобы не засорять дисковое пространство;

5) сервис администратора должен позволять просматривать данные, полученные от локального клиента конкретного сотрудника;

6) сервис администратора должен позволять удалять данные о конкретных сотрудниках;

7) локальный клиент должен устанавливаться на компьютер сотрудника;

8) локальный клиент должен настраиваться путем передачи параметров через консоль, либо вручную, с помощью редактирования файла конфигурации в папке с установленным локальным клиентом;

9) локальный клиент должен автоматически запускаться со стартом компьютера сотрудника;

10) локальный клиент должен получать скриншоты и список процессов с компьютера сотрудника;

11) локальный клиент должен проверять список процессов сотрудника на подозрительные программы;

12) локальный клиент должен отправлять полученные данные на сервер.

2.1.2. Нефункциональные требования

Нефункциональные требования описывают свойства и ограничения, накладываемые на информационную систему. Нефункциональные требования определяют бизнес-правила, системные требования и общие требования к ПО. Для создания клиент-серверного приложения для контроля сотрудников за работой на компьютере были определены следующие нефункциональные требования:

1) система должна быть написана на языке Python с использованием Django Framework, а также дополнительных библиотек;

2) сервис администратора должен быть доступен на всех современных устройствах с выходом в интернет;

3) сервис администратора должен быть доступен только администратору;

4) локальный клиент должен вести себя скрытно от пользователя и не обращать на себя внимания;

5) локальный клиент должен работать на операционных системах семейства Windows 7 и выше.

2.2. Варианты использования системы

С помощью языка для объектного моделирования UML [9] были построены диаграммы вариантов использования клиент-серверного приложения для контроля сотрудников за работой на компьютере, изображенные на рисунке 1 и рисунке 2.

С сервером взаимодействуют два актера, один из которых является гостем, а другой – администратором (рис. 1.).

Гость – это человек, посетивший сайт сервера системы. Гость является неавторизованным пользователем без доступа к какому-либо функционалу сервиса, кроме возможности авторизации.

Администратор – это суперпользователель, который может просматривать поступившую от локальных клиентов информацию, добавлять, удалять других администраторов и изменять их данные, удалять сотрудников из системы.

С компьютером сотрудника взаимодействует всего один актер – локальный клиент (рис. 2.).

Локальный клиент представляет собой запущенный на компьютере конкретного сотрудника исполняемый файл. Он может принимать параметры и автоматически устанавливаться на компьютер сотрудника, отправлять скриншоты и список процессов работника.

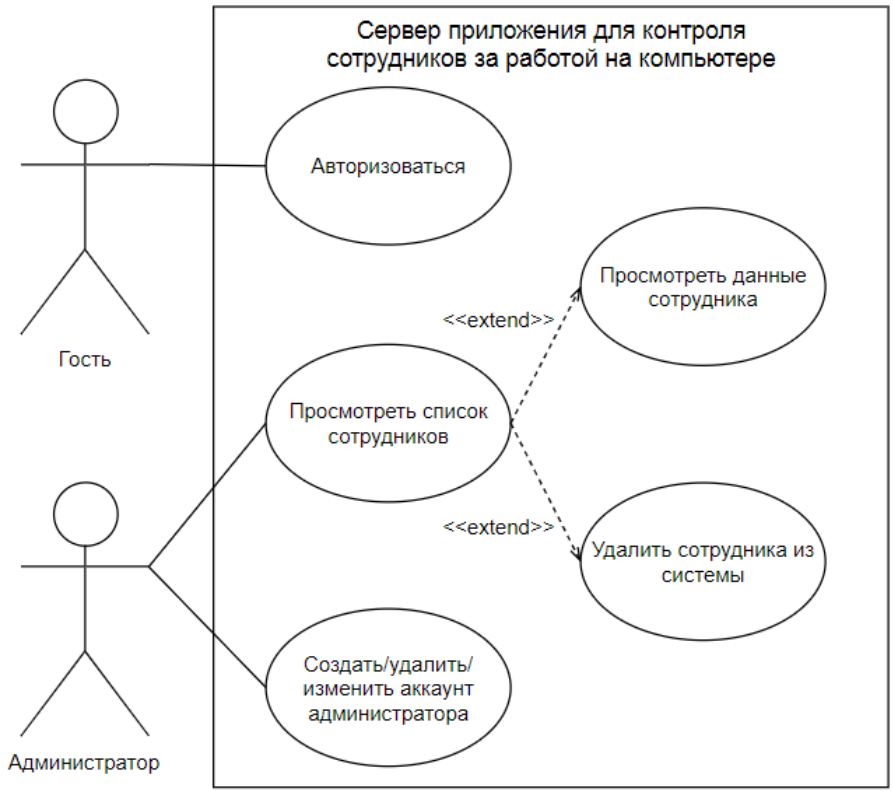

Рис. 1. Диаграмма вариантов использования сервера

Представленные на диаграмме с первого рисунка актеры совершают следующие действия.

Гость может *авторизоваться* введя логин и пароль администратора на странице авторизации.

Администратор может:

- *создать/удалить/изменить аккаунт администратора* – с помощью сервиса администратора создавать новых администраторов, удалять существующих или вносить в них изменения;
- *просмотреть список сотрудников* – переходить на страницу со списком всех сотрудников, зарегистрированных в системе;
- *просмотреть данные сотрудника* – переходить на страницу обзора данных, полученных от локального клиента на компьютере сотрудника;

- *удалить сотрудника из системы* – удалять все записи о сотруднике с сервера, таким образом исключив дальнейшее появление данных с его локального клиента.

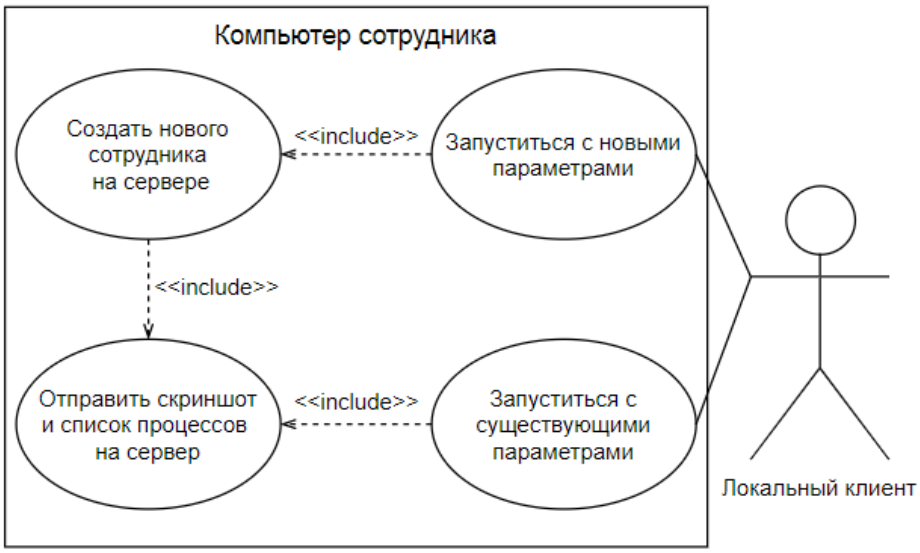

Рис. 2. Диаграмма вариантов использования компьютера сотрудника локальным клиентом

На рисунке 2 изображена диаграмма деятельности, на которой локальный клиент взаимодействует с компьютером сотрудника.

Локальный клиент может:

- *запуститься с новыми параметрами* – запуститься, используя введенные ключи, таким образом изменить текущую конфигурацию, либо создать новую;
- *запуститься с существующими параметрами* – запуститься, используя уже известные клиенту параметры и продолжить работу;
- *создать нового сотрудника на сервере* – создать новую запись о сотруднике на сервере;
- *отправить скриншот и список процессов на сервер* – создать новую запись в каталоге сотрудника на сервере и внести в нее скриншот и список процессов.

2.3. Архитектура системы

Для статического изображения архитектуры клиент-серверного приложения для контроля сотрудников за работой на компьютере была использована диаграмма компонентов.

Диаграмма компонентов наглядно демонстрирует разбиение программной системы на структурные компоненты и связи (зависимости) между ними.

На рисунке 3 изображена диаграмма компонентов системы, выполненная в нотации UML [9]. Диаграмма состоит из следующих компонентов:

- сервер системы для контроля сотрудников;
- сайт системы для контроля сотрудников;
- локальный клиент на компьютере сотрудника;
- конфигурационный файл config.ini на компьютере сотрудника.

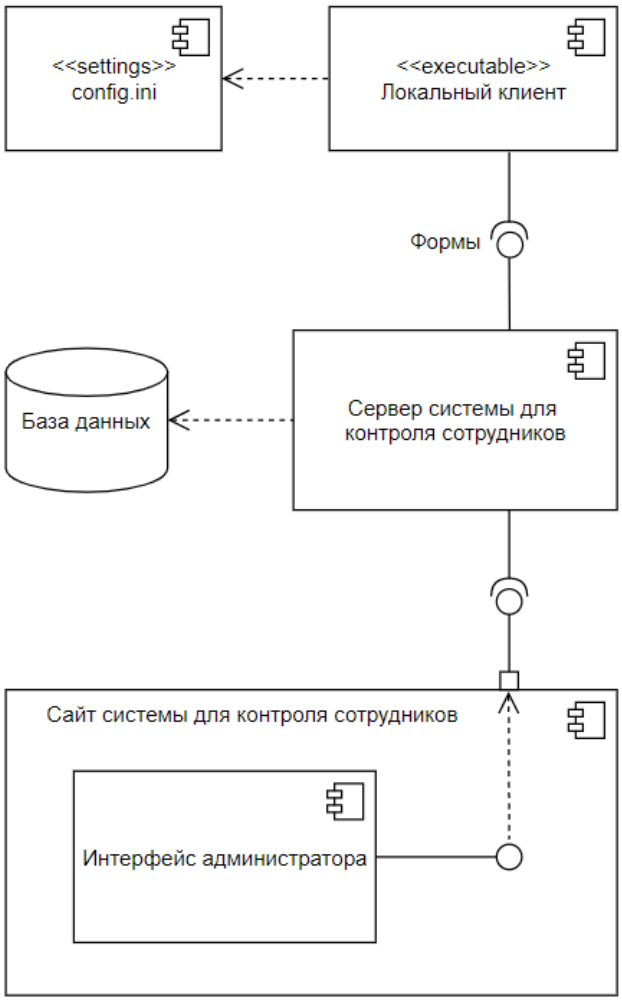

Рис. 3. Диаграмма компонентов системы для контроля сотрудников

Сервер системы для контроля сотрудников хранит информацию об администраторах, сотрудниках, включая изображения, полученные от локальных клиентов. Сервер реализует многие функции обработки данных, включая обработку форм, автоматическое удаление устаревших записей, проверку секретного ключа, пагинацию и т.д.

Сайт системы для контроля сотрудников предоставляет интерфейс администратора системы. Интерфейс администратора предоставляет функции добавления, изменения, удаления аккаунтов администраторов. Интерфейс позволяет просматривать список внесенных в базу данных сотрудников, удалять их. И, что самое главное, он позволяет просматривать страницы отслеживания конкретных работников. Интерфейс администратора формируется с использованием стандартного подхода к созданию административного интерфейса, предоставляемого платформой Django [11]. Доступ к интерфейсу имеют только администраторы.

Взаимодействие между локальным клиентом и сервером обеспечивают стандартные *формы* Django Forms, которые имеют широкие возможности настройки и применения. В нашем случае используется две формы: одна – для создания новых сотрудников, вторая – для получения данных о конкретном сотруднике от локального клиента.

Локальный клиент системы постоянно взаимодействует с сервером и выполняет важную роль в системе контроля сотрудников. Он обеспечивает возможность собственной установки, автоматического запуска со стартом компьютера, постоянной и скрытной работы, создания и изменения файла конфигурации, режима ожидания, автоматического переподключения, анализа запущенных процессов, создания скриншотов и отправку процессов и скриншотов на сервер через установленный промежуток времени.

config.ini – это конфигурационный файл, который создает локальный клиент при первом запуске, записывая в него настройки, и который в дальнейшем может перезаписываться с помощью клиента или в ручном режиме.

Он хранит такую информацию как имя сотрудника, частоту обновления и список подозрительных процессов.

2.4. Схема базы данных сервера системы

Исходя из требований к системе была разработана схема базы данных, представленная на рисунке 4. Схема выполнена в нотации Crow's Foot [1]. Также на этапе проектирования были описаны названия всех таблиц и полей базы данных.

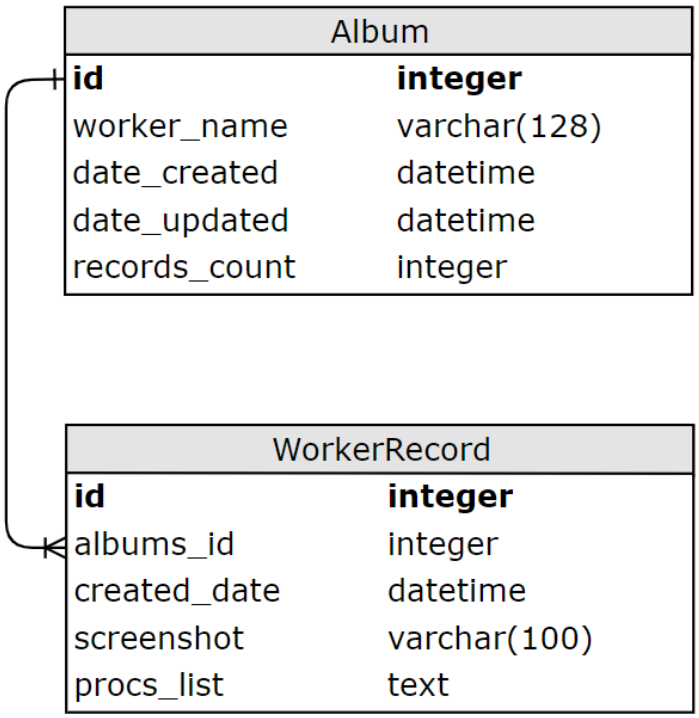

Рис. 4. Схема базы данных

Опишем таблицы и поля базы данных.

Таблица *WorkerRecord* содержит информацию об отправленных с компьютеров сотрудников записях:

- *id* – поле уникального идентификатора с автоматическим инкрементом;
- *albums_id* – поле внешнего ключа, уникальный идентификатор альбома сотрудника, к которому относится запись в таблице;
- *created_date* – поле времени и даты создания записи в базе данных;

- *screenshot* – поле названия файла скриншота в файловом хранилище на сервере;
- *procs_list* – поле списка процессов, в том числе проанализированных и помеченных (в случае, если он указан в конфигурационном файле локального клиента как подозрительный), запущенных на компьютере сотрудника.

Таблица Album содержит в себе информацию о сотрудниках и о их альбоме:

- *id* – поле уникального идентификатора с автоматическим инкрементом;
- *worker_name* – поле имени сотрудника;
- *date_created* – поле времени и даты создания альбома сотрудника на сервере;
- *date_updated* – поле времени и даты последней поступившей в альбом сотрудника записи;
- *records_count* – поле количества записей, внесенных в альбом сотрудника.

Вывод

В данной главе были составлены и описаны как функциональные, так и нефункциональные требования к разрабатываемому клиент-серверному приложению для контроля сотрудников за работой на компьютере. Были смоделированы диаграмма вариантов использования системы, диаграмма компонентов системы и схема базы данных системы.

3. РЕАЛИЗАЦИЯ

3.1. Выбор средства разработки сервера

Для разработки серверной части системы был выбран фреймворк Django, так как на сегодняшний день именно этот фреймворк является наиболее популярным среди разработчиков (как новичков, так и профессионалов) [13].

Django – свободный фреймворк для создания веб-приложений, написанный на языке Python и использующий шаблон проектирования, подобный шаблону MVC.

В отличие от некоторых других фреймворков, обработчики URL в Django указываются явно при помощи регулярных выражений, а не строятся автоматически из структуры моделей контролеров [17].

Для работы с базами данных Django использует собственный ORM, в котором модель представления данных описывается Python классами, а по ней уже генерируется сама схема таблиц базы данных.

Django позволяет создавать динамические веб-приложения за довольно небольшой промежуток времени. Платформа Django спроектирована таким образом, чтобы разработчики смогли сосредоточиться на решении содержательных задач, а не отвлекаться на реализацию тривиальных потребностей (аутентификации и авторизации пользователей, настройки работы с базой данных и др.). Для достижения этой цели Django предоставляет общеупотребительные шаблоны веб-разработки высокого уровня абстракции, инструменты для быстрого выполнения часто встречающихся задач программирования и четкие соглашения о способах решения проблем [10].

Архитектура Django схожа с шаблоном «Модель-Представление-Контроллер» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (англ. View), а презентационная логика представлений реализуется в Django уровнем Шаблонов (англ. Template). Из-за этого уровневую архитектуру Django можно назвать «Модель-Шаблон-Представление» (MTV).

Сервисы, построенные на фреймворке Django, при получении запроса на определенный url обращается к файлу привязки URL, в Django сервисах он имеет стандартное название urls.py, где .py расширение программных модулей, написанных на Python.

Модуль urls.py можно рассматривать как таблицу с содержанием нашего сайта. Проще говоря, этот файл определяет соответствие между URL и функциями представления (из модуля views.py), которые должны быть вызваны для этих URL. Именно так мы указываем Django, какой код необходимо выполнить для каждого запрошенного URL.

Модуль views.py содержит функции представлений. Основная идея в том, что каждое представление является функцией языка Python, которая и отвечает за формирование HTTP-ответа и HTML-страниц для веб-сервера. Для того, чтобы сформировать такой ответ, представления используют HTML-шаблоны из папки с шаблонами, модели из файла models.py, формы из файла forms.py и данные из базы данных, к которой подключается сервер.

Для разметки страниц административной части системы используется язык HTML. HTML (HyperText Markup Language) – стандартизированный язык разметки веб-страниц во Всемирной паутине. Большинство веб-страниц в интернете содержат описание разметки на языке HTML.

3.2. Выбор средства разработки клиента

Для разработки клиентской части системы для контроля сотрудников за работой на компьютере был выбран язык Python, так как Django, в свою очередь, тоже основан на Python. Их совместное использование позволит добиться максимально полного понимания принципов работы этого языка и использования лучших его возможностей.

Python – это высокоуровневый универсальный мультипарадигмальный язык программирования с открытым исходным кодом. Python поддерживает такие парадигмы программирования, как объектно-ориентированная, функ-

циональная и процедурная. Он обычно используется как для создания автономных программ, так и для написания узкоспециализированных скриптов в самых разных областях, и, как правило, считается одним из наиболее широко используемых языков программирования в мире. Стандартный набор библиотек Python включает большое количество полезных функций [3].

Основными архитектурными особенностями языка Python являются динамическая типизация, автоматическое управление памятью, механизм обработки исключений, поддержка многопоточной работы и удобные высокоуровневые структуры данных. Все это также является преимуществом при выборе языка программирования.

Для создания скриншотов рабочего стола компьютеров, используемых сотрудниками будет использоваться сторонний пакет Pillow [5], который позволяет работать с графикой.

Для получения списка запущенных на компьютере сотрудника процессов будет использоваться сторонний пакет Psutil [6], который позволяет получать разнообразные данные о запущенных на компьютере процессах и обрабатывать эти данные.

Для создания конечного исполняемого файла с расширением .exe (для операционных систем семейства Windows) будет применен сторонний пакет PyInstaller [7], который способен упаковывать код, написанный на Python, со всеми необходимыми библиотеками, включенными зависимостями и сторонними пакетами в единственный исполняемый файл. Этот файл в дальнейшем будет играть роль клиента в нашей системе.

3.3. Реализация серверной части

В рамках создания серверной части приложения для контроля сотрудников за работой на компьютере необходимо реализовать структуру базы данных, внутреннюю логику обработки данных на сервере и внешний вид страниц административного сайта.

3.3.1. Реализация структуры базы данных

Реализация серверной части началась с реализации структуры базы данных. В рамках фреймворка Django правильным подходом для решений этой задачи является использование классов-моделей механизма миграции. Механизм миграций решет проблему того, что структура БД во время разработки приложения может изменяться, но это не отражается в исходном коде приложения.

В качестве СУБД была использована SQLite, сама база данных представлена в системе одним файлом db.sqlite3.

В соответствии со схемой базы данных, приведенной в разделе проектирование, мы создаем в файле *models.py*, отвечающий за представление данных в базе данных, два класса-модели – Album и WorkerRecord. Каждый из них будет содержать нужные нам поля, ограничения для них и представлять одноименную таблицу в базе данных. Код классов-моделей Album и WorkerRecord приведен в листинге 1.

Листинг 1. Создание таблиц в базе данных

```
class Album(models.Model):
    worker_name = models.CharField("Имя сотрудника", max_length=128)
    date_created = models.DateTimeField("Дата добавления",
                                        default=timezone.now)
    date_updated = models.DateTimeField("Дата обновления",
                                        default=timezone.now)
    records_count = models.IntegerField("Кол-во записей", default=0)

    class Meta:
        verbose_name = 'альбом сотрудника
        verbose_name_plural = 'Сотрудники'

class WorkerRecord(models.Model):
    created_date = models.DateTimeField("Дата добавления",
                                        default=timezone.now)
    screenshot = models.FileField("Скриншот")
    albums = models.ForeignKey(Album, on_delete=models.CASCADE,
                               blank=True)
    procs_list = models.TextField("Список процессов")
```

3.3.2. Реализация внутренней логики

Интерфейсом, используемым для приема данных от локальных клиентов, стали стандартные формы Django Forms, автоматически генерируемые по существующим классам-моделям и указанным полям. Одна из форм необходима для создания новых альбомов сотрудников, другая – для добавления новых записей в альбомы. Код форм находится в файле *forms.py*.

В функции представления для обоих форм файла *views.py* был добавлен код, ограничивающий несанкционированное внесение новых записей в базу данных кем-то, помимо локального клиента. Также, подобный код был добавлен в модуль генерации форм *forms.py* для дополнительной безопасности. Авторизация осуществляется введением секретного ключа. Код приведен на листинге 2, где `secret_key` – заданный на сервере секретный ключ, а `input_key` – полученный от клиента секретный ключ.

Листинг 2. Проверка ключа

```
#views.py
input_key = form.cleaned_data.get('key')
if input_key == secret_key:
      pass
else:
      return render(request, 'form.html', context)
```

В представлении формы для создания новых записей о сотрудниках, с помощью объектно-реляционного отображения Django ORM, при добавлении новой записи производится запрос в базу данных на обновление полей `date_updated` и `records_count`. Код представлен в листинге 3.

Листинг 3. Обновление даты загрузки и числа записей

```
#views.py
new_date = WorkerRecord.objects.filter(albums=albums).last().created_date
Album.objects.filter(worker_name=albums).update(date_updated=new_date)

new_count = WorkerRecord.objects.filter(albums=albums).count()
Album.objects.filter(worker_name=albums).update(records_count=new_count)
```

Максимальный объем памяти, занимаемый одним скриншотом, примерно равняется 250 килобайтам. Если предположить, что в компании работает 50 сотрудников с установленными и настроенными на их компьютерах локальными клиентами, с десятичасовым рабочим днем и двумя выходными в неделю, а частота работы клиента равна 1 скриншот в минуту, то за месяц будет израсходовано около $\frac{250 \times 60 \times 10 \times 20 \times 50}{1024 \times 1024} \approx 143$ гигабайт, что является приемлемым объемом памяти для современных компьютеров, а тем более серверов. Поэтому было решено добавить функцию автоматического удаления записей и скриншотов старше 30 дней из базы данных и файлового хранилища. Код представлен в листинге 4, где `remove_time` – устанавливаемая константа, определяющая максимальный срок хранения записи и ее скриншота.

Листинг 4. Автоматическое удаление старых скриншотов и записей

```
#views.py
remove_time = 30
delta = timedelta(days=remove_time)
last_date_time = WorkerRecord.objects.all().last().created_date
delete_date_time = last_date_time - delta
oldRecords = WorkerRecord.objects.filter(created_date__lte=de-
lete_date_time)
old_img_names = oldRecords.values_list('screenshot', flat=True)
for image in old_img_names:
      try:
            WorkerRecord.objects.filter(screenshot=image).delete()
            os.remove(os.path.join(settings.MEDIA_ROOT, image))
      except FileNotFoundError:
            print("Exception, file " + image + " not found.")
```

В процессе разработки выяснилось, что доступ по прямой ссылке на скриншоты рабочих столов сотрудников на сервере имеется не только у авторизованного администратора, а у всех, кто попытается перейти по ней. Для защиты файлов была написана функция, привязанная к URL и представленная в листинге 5. Теперь, при переходе по прямому пути расположения

скриншота рабочего стола сотрудника на сервере без авторизации, происходит перенаправление на страницу авторизации.

Листинг 5. Защита скриншотов от несанкционированного доступа

```
#urls.py
urlpatterns = [...,
               url(r'^{}(?P<path>.*)$'.format(settings.MEDIA_URL[1:]),
               views.protected_serve, {'document_root':
               settings.MEDIA_ROOT}),
               ...,]
#views.py
@login_required
def protected_serve(request, path, document_root=None):
    try:
        image_url = WorkerRecord.objects.filter(screenshot=path)
                    .values_list('screenshot', flat=True)[0]
        correct_image_url = image_url.replace("/", "")
        if correct_image_url == path:
            return serve(request, path, document_root)
    except ObjectDoesNotExist:
        return HttpResponse("Sorry, you don't have permission to access
                             this file")
```

3.3.3. Реализация интерфейса администратора

Стандартный интерфейс администратора для добавления, изменения и удаления объектов в/из базы данных предоставляется платформой Django. Интерфейс администратора, необходимый для наших задач, должен делать несколько больше, чем просто взаимодействовать с базой данных. Поэтому стандартный административный интерфейс был изменен под условия нашего сервиса, путем добавления новых полей и изменений в исходные коды административной части платформы Django.

Для того, чтобы администратор мог с легкостью ориентироваться в статистике по сотрудникам и в их галереях, была модифицирована страница представления таблицы базы данных WorkerRecord. Внешний вид этой административной страницы представлен на рисунке 5.

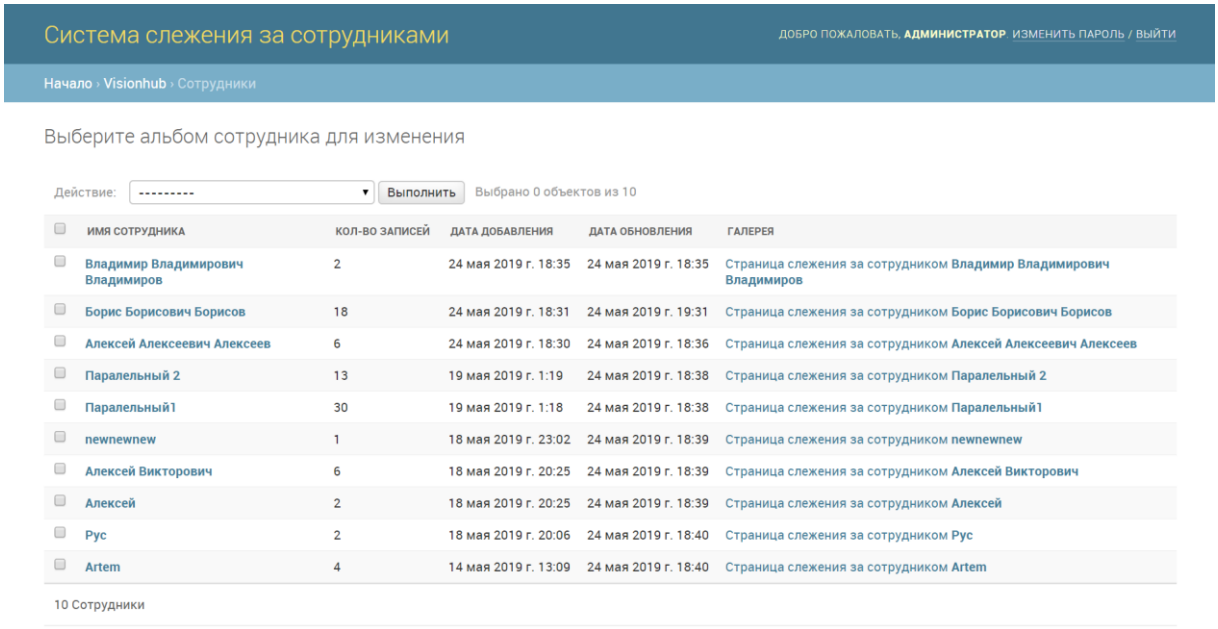

Рис. 5. Страница альбомов сотрудников

Страница слежения за конкретным сотрудником представляет из себя html страницу, в шаблон (template) которой, с помощью представления из файла *views.py* передается список записей о сотруднике. Также, для быстрой загрузки страницы и решения проблемы показа всех существующих записей пользователя, была реализована пагинация. Код указан в листинге 6.

Листинг 6. Функция отображения страницы слежения за сотрудником

```
def worker(request, worker_name):
    workerAlbumSet = Album.objects.filter(worker_name=worker_name)
    workerAlbumIdSet = workerAlbumSet.values_list('id', flat=True)
    albumId = workerAlbumIdSet[0]
    snapsByWorker = list(reversed(WorkerRecord.objects.filter
                                  (albums_id=albumId)))
    paginator_snaps = Paginator(snapsByWorker, 10)
    page = request.GET.get('page')
    try:
        snapsByWorker = paginator_snaps.get_page(page)
    except (EmptyPage, PageNotAnInteger):
        snapsByWorker = paginator_snaps.get_page(pagina-
tor_snaps.num_pages)
    return render(request, 'workerInfo.html', {'snaps': snapsByWorker,
                                               'album': workerAlbumSet})
```

Таким образом, передавая данные в созданный нами html шаблон, Django автоматически генерирует страницу слежения за конкретным сотрудником. Рисунок 6 демонстрирует, как выглядит эта страница. В названии и в верхней части страницы указано имя сотрудника, записи которого отображаются в отдельных блоках далее на странице. Каждый такой блок состоит из даты и времени создания записи, скриншота и списка запущенных процессов, в котором отмечены подозрительные программы. Также, вверху и внизу страницы отслеживания имеются блоки навигации по страницам.

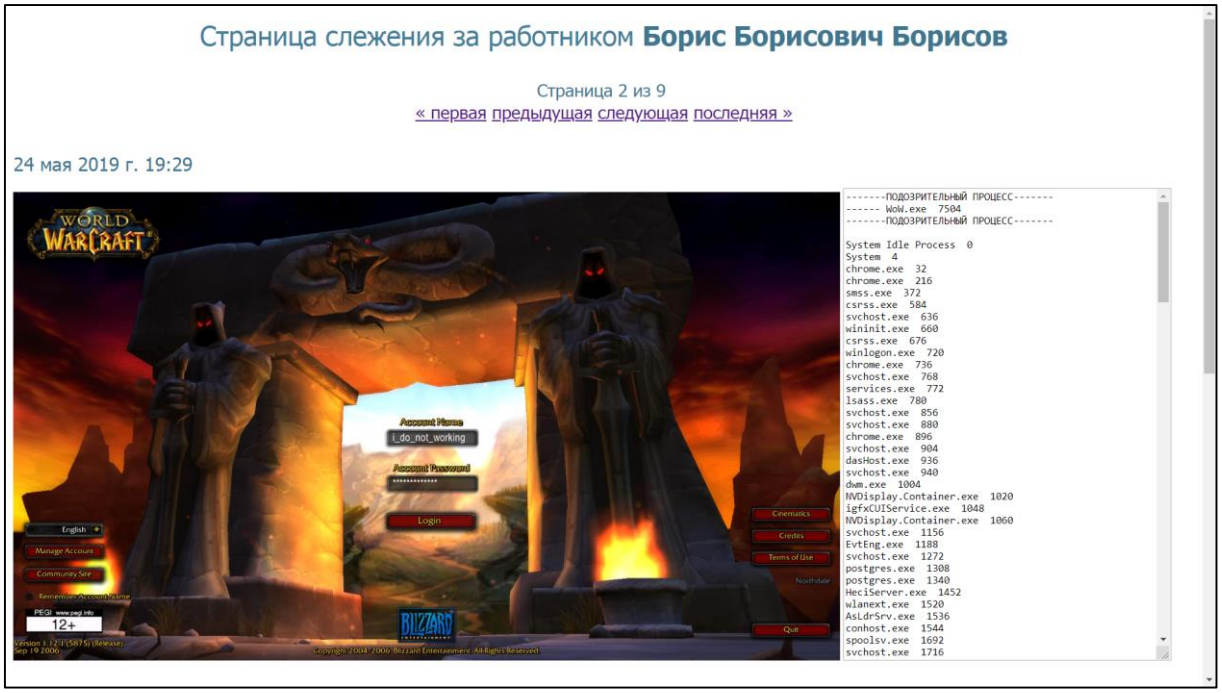

Рис. 6. Страница слежения за сотрудником

3.4. Реализация клиента

В рамках реализации клиентской части приложения для контроля сотрудников за работой на компьютере, необходимо создать исполняемый файл, который будет принимать настройки, устанавливаться на компьютер сотрудника и отсылать данные на сервер все то время, что запущен наблюдаемый ПК и доступен сервер.

В первую очередь, необходимо объявить и получить параметры, которые будут использоваться при установке или настройке клиента. Необходимые для установки аргументы, это имя сотрудника, время обновления, указанное в минутах, и путь установки локального клиента. Имени соответствует ключ `-n`, времени обновления – ключ `-t`, а пути установки – ключ `-p`. Все значения вводятся через пробел после ключа, а имя и путь установки – в кавычках. В листинге 7 указано, как они объявляются и получаются.

Листинг 7. Создание параметров запуска

```
parser = argparse.ArgumentParser()
parser.add_argument("-n", "--name", action="store")
parser.add_argument("-t", "--time", action="store")
parser.add_argument("-p", "--path", action="store")
args = parser.parse_args()

worker = args.name
updateTime = args.time
path = args.path
```

В случае, если ключей с аргументами при запуске не поступило, то локальный клиент попытается проверить, существует ли непустой конфигурационный файл *config.ini* в том же каталоге, что и исполняемый файл клиента. При обнаружения непустого конфигурационного файла, клиент считает из него настройки и продолжит работу в штатном режиме. При первой установке, клиент создает конфигурационный файл и записывает в него имя сотрудника, время обновления и стандартный список подозрительных процессов. В дальнейшем присутствует возможность вручную изменять настройки в файле *config.ini*, в том числе добавлять названия новых процессов в список подозрительных.

В ситуации первого запуска, локальный клиент скопирует себя по пути установки и добавит запись в реестр Windows, которая обеспечит автоматический запуск программы при старте компьютера. Код добавления записи в реестр продемонстрирован в листинге 8.

Листинг 8. Создание записи в реестре Windows

```
key = winreg.OpenKey(winreg.HKEY_CURRENT_USER,
                     'Software\Microsoft\Windows\CurrentVersion\Run', 0,
                      winreg.KEY_ALL_ACCESS)
winreg.SetValueEx(key, 'WCSStartup', 0, winreg.REG_SZ,
                  PATH.replace(r'/', '\\') +
                  os.path.basename(sys.argv[0]))
key.Close()
```

Следующим шагом является проверка наличия сотрудника со считанным из конфигурационного файла именем в базе данных и, если выявлено его отсутствие в базе, создание нового альбома сотрудника. Все это сопровождается циклом переподключения к серверу, так как он может быть недоступен, либо на компьютере сотрудника в данный момент может отсутствовать подключение к интернету. В коде листинга 9, где `exists_albums` – список существующих сотрудников, а `albumUrl` – адрес страницы с формой создания нового альбома, проводится попытка подключения и создания нового альбома сотрудника, в случае его отсутствия в списке `exists_albums`.

Листинг 9. Создание нового альбома сотрудника

```
while True:
      try:
            send_form_html_connect = urllib.request.urlopen
                                     ("http://127.0.0.1:8000/hub/")
            send_form_html_bytes = send_form_html_connect.read()
            send_form_html_text = send_form_html_bytes.decode("utf8")
            send_form_html_connect.close()
            exists_albums = re.findall('<option value="(.*)">',
                                       send_form_html_text)
            if worker in exists_albums:
                  pass
            else:
                  r = requests.post(albumUrl, data=payload)
                  print(r.status_code, "Album created", worker)
            break
      except (requests.exceptions.RequestException,
              requests.exceptions.ConnectionError, urllib.error.URLError):
            time.sleep(10)
```

Последним шагом работы локального клиента является штатная работа программы, при которой действует бесконечный цикл с паузой, равной времени обновления в минутах. Этот цикл создает и сохраняет скриншот в папку локального клиента, получает список запущенных на компьютере сотрудника процессов и помечает подозрительные. Далее, клиент пытается отправить данные и в случае отсутствия подключения удаляет сохраненный скриншот и повторяет вышеописанный алгоритм через 10 секунд. Данные состоят из имени сотрудника, файла скриншота рабочего стола, списка процессов и секретного ключа для аутентификации. Секретный ключ и таймер переподключения, установленный на 10 секунд, также использует алгоритм создания нового альбома. В листинге 10 показано, как создается новый скриншот, как помечается список процессов, а также как эта информация отсылается на сервер. Следует дополнить, что каждый раз после загрузки скриншота на сервер, файл удаляется с компьютера сотрудника.

Листинг 10. Получение и отправка данных на сервер

```
snapshot = ImageGrab.grab()
imgSavePath = PATH + "snap_" + worker + "_" +
              str(int(time.time())) + ".jpg"
parsed_processes = ''
for proc in psutil.process_iter():
      if proc.name().lower() in processes:
            parsed_processes = ''.join(
            ("-------ПОДОЗРИТЕЛЬНЫЙ ПРОЦЕСС-------\n------ " +
             str(proc.name()) + "  " + str(proc.pid) +
             "\n-------ПОДОЗРИТЕЛЬНЫЙ ПРОЦЕСС-------\n" +
             '\n', parsed_processes))
      else:
            parsed_processes = parsed_processes + str(proc.name()) +
                               "  " + str(proc.pid) + '\n'

payload = {'worker_name': worker, 'albums': worker,
           'procs_list': parsed_processes, 'key': secret_key}
snapshot.save(imgSavePath)
f = open(imgSavePath, 'rb')
r = requests.post(urls, files={'screenshot': f}, data=payload)
```

Для того, чтобы наша программа работала на компьютерах, на которых не установлен Python, создадим исполняемый *.exe* файл с помощью стороннего пакета PyInstaller. Он автоматически включит в исполняемый файл все библиотеки, необходимые для корректного функционирования программы. Команда *pyinstaller --icon "icon.ico" --noconsole --onefile csrss.py* означает следующее: --icon – указывает на файл и иконкой будущего файла, в данном случае выбрана стандартная иконка исполняемого файла в Windows, *--noconsole* – параметр указывающий на то, чтобы будущий исполняемый файл запускался без вызова консоли, тем самым маскируя работу клиента, *--onefile* – означает упаковку всех библиотек в исполняемый файл без добавления их в его директорию и, наконец, название файла *csrss.exe* – есть ни что иное, как имя стандартной службы Windows, без которой функционирование системы невозможно. Это название дополнительно замаскирует от сотрудника работу локального клиента. После введения команды, получаем рабочий исполняемый файл локального клиента, изображенный на рисунке 7.

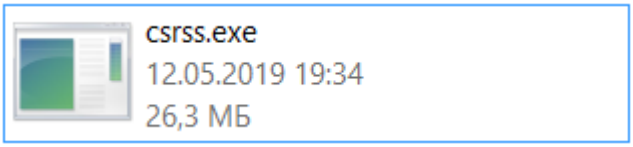

Рис. 7. Локальный клиент

Объем памяти, занимаемый клиентом, составил чуть более 26 мегабайт.

Вывод

В третьей главе были выбраны и описаны инструменты для разработки. Были реализованы серверная часть, включая структуру базы данных, внутреннюю логику и интерфейс администратора, а также клиентская часть клиент-серверного приложения для контроля сотрудников за работой на компьютере.

4. ТЕСТИРОВАНИЕ

Разработанное веб-приложение прошло функциональное тестирование. Функциональное тестирование – это тестирование программного обеспечения с целью проверки соблюдения функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, поставленные перед ней на этапе проектирования. Результаты тестирования приведены в таблице 2. Проведены необходимые тесты, охватывающие весь функционал разработанного клиент-серверного приложения.

Табл. 2. Функциональное тестирование

№ п/п	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1.	Авторизация	1. Пользователь заходит на страницу авторизации. 2. Вводит логин и пароль. 3. Нажимает кнопку «Войти».	Пользователь авторизуется, попадает в интерфейс администратора и получает его права.	Да
2.	Добавление нового администратора	1. Администратор заходит на страницу списка администраторов. 2. Администратор нажимает кнопку «Добавить». 3. Администратор вводит новые логин и пароль и нажимает кнопку «Сохранить».	Новый администратор создан, администратор перенаправляется на страницу редактирования пользователя.	Да

Продолжение табл. 2

№ п/п	Название теста	Шаги	Ожидаемый результат	Тест пройден?
3.	Удаление администратора	1. Администратор заходит на страницу списка администраторов. 2. Администратор выбирает одного или нескольких пользователей из списка. 3. Администратор выбирает «Удалить», а затем «Выполнить» и подтверждает действие.	Выбранные администраторы удалены, администратор перенаправляется на ту же страницу.	Да
4.	Удаление сотрудников	1. Администратор заходит на страницу просмотра альбомов сотрудников. 2. Администратор выбирает одного или нескольких сотрудников из списка. 3. Администратор выбирает «Удалить», а затем «Выполнить» и подтверждает действие.	Выбранные альбомы сотрудников и связанные с ними записи удалены, администратор перенаправляется на ту же страницу.	Да

Продолжение табл. 2

№ п/п	Название теста	Шаги	Ожидаемый результат	Тест пройден?
5.	Просмотр данных конкретного сотрудника	1. Администратор заходит на страницу просмотра альбомов сотрудников. 2. Администратор нажимает на ссылку, ведущую на страницу отслеживания сотрудника.	Происходит переход на страницу с данными сотрудника. Данные отображаются корректно и упорядочено.	Да
6.	Переход по страницам в альбоме сотрудника	1. Администратор заходит на страницу просмотра альбомов сотрудников. 2. Администратор нажимает на ссылку, ведущую на страницу отслеживания сотрудника. 3. Администратор нажимает на ссылку другой страницы.	Происходит перенаправление на другую страницу и отображение более новых или более старых данных сотрудника.	Да
7.	Запрет доступа к скриншотам по ссылке без авторизации	1. Администратор нажимает на кнопку «Выйти». 2. Неавторизованный пользователь	Пользователь перенаправляется на страницу авторизации.	Да

Продолжение табл. 2

№ п/п	Название теста	Шаги	Ожидаемый результат	Тест пройден?
		переходит по ссылке на файл.		
8.	Автоматическое удаление устаревших скриншотов и записей	1. Сервер получает новые данные от локального клиента 2. Определяются записи и скриншоты, старше времени загрузки новых данных на 30 дней. 3. Записи и скриншоты старше 30 дней удаляются.	Новые данные получены, старые записи и скриншоты удалены.	Да
9.	Установка локального клиента	1. Администратор запускает локальный клиент с заданными параметрами 2. Локальный клиент устанавливается, добавляет запись в реестр, создает конфигурационный файл. 3. Локальный клиент создает на сервере новый альбом сотрудника.	Клиент скопирован в папку на компьютере, в реестр добавлена запись для автозапуска, на сервере создан альбом нового сотрудника.	Да

Продолжение табл. 2

№ п/п	Название теста	Шаги	Ожидаемый результат	Тест пройден?
9.	Установка локального клиента	1. Администратор запускает локальный клиент с заданными параметрами 2. Локальный клиент устанавливается, добавляет запись в реестр, создает конфигурационный файл. 3. Локальный клиент создает на сервере новый альбом сотрудника.	Клиент скопирован в папку на компьютере, в реестр добавлена запись для автозапуска, на сервере создан альбом нового сотрудника.	Да
10.	Штатная работа локального клиента	1. Локальный клиент запускается. 2. Локальный клиент считывает настройки из файла конфигурации. 3. Локальный клиент отправляет данные на сервер.	Данные получены сервером и отображаются на странице сотрудника.	Да
11.	Параллельная работа не-	1. Запускается несколько локальных	Сервер принимает и отображает данные со	Да

Продолжение табл. 2

№ п/п	Название теста	Шаги	Ожидаемый результат	Тест пройден?
	скольких локальных клиентов	клиентов с разными параметрами. 2. Локальные клиенты отправляют данные на сервер.	всех запущенных локальных клиентов.	
11.	Параллельная работа нескольких локальных клиентов	1. Запускается несколько локальных клиентов с разными параметрами. 2. Локальные клиенты отправляют данные на сервер.	Сервер принимает и отображает данные со всех запущенных локальных клиентов.	Да
12.	Работа в режиме ожидания	1. Клиент отправляет данные. 2. Сервер принимает данные. 3. Сервер становится недоступен. 4. Клиент входит в режим ожидания и ничего не отправляет. 5. Сервер становится доступен.	На время, что недоступен сервер, клиент продолжает работу и не отправляет данные. При появлении соединения работа восстанавливается.	Да

Продолжение табл. 2

№ п/п	Название теста	Шаги	Ожидаемый результат	Тест пройден?
		6. Клиент восстанавливает работу.		
13.	Запуск при старте Windows	1. ОС Windows запускается. 2. Локальный клиент запускается. 3. Локальный клиент начинает работу.	Локальный клиент запускается вместе с системой и работает в штатном режиме.	Да

Вывод

В данной главе было проведено функциональное тестирование клиент-серверного приложения для контроля сотрудников за работой на компьютере. Все тесты показали, что система функционирует согласно требованиям.

ЗАКЛЮЧЕНИЕ

В процессе выполнения работы по разработке клиент-серверного приложения для контроля сотрудников за работой на компьютере были решены следующие задачи:

1) был выполнен анализ предметной области;
2) было спроектировано клиент-серверное приложения;
3) было реализовано клиент серверное приложение;
4) было проведено тестирование клиента, сервера и их в совокупности.

Во время выполнения выпускной квалификационной работы были изучены возможности разработки серверной части приложений, а также особенности работы с операционной системой Windows в качестве платформы для программного взаимодействия.

Разработанная архитектура клиент-серверного приложения позволит в будущем развивать его и оснащать новым функционалом и возможностями.

ЛИТЕРАТУРА

1. Coronel C., Morris S. Database systems: design, implementation, & management. – USA: Cengage Learning, 2016. – 752 p.

2. Employees Really Do Waste Time at Work. [Электронный ресурс] URL: https://www.forbes.com/sites/cherylsnappconner/2012/07/17/employees-really-do-waste-time-at-work/ (дата обращения: 29.04.2019).

3. Lutz M. Python Pocket Reference: Python In Your Pocket. – USA: O'Reilly Media, Inc., 2014. – 254 p.

4. Organisation for Economic Co-operation and Development. Directorate for Science, Technology, and Industry. OECD Science, Technology and Industry Scoreboard 2015: Innovation for Growth and Society. – OECD, 2015.

5. Pillow (PIL Fork) 6.0.0 documentation. [Электронный ресурс] URL: https://pillow.readthedocs.io/en/stable/ (дата обращения: 03.05.2019).

6. Psutil 5.6.3 documentation. [Электронный ресурс] URL: https://psutil.readthedocs.io/en/latest/ (дата обращения: 03.05.2019).

7. PyInstaller Quickstart. [Электронный ресурс] URL: https://www.pyinstaller.org/ (дата обращения: 04.05.2019).

8. Автоматический учет рабочего времени CrocoTime. [Электронный ресурс] URL: https://crocotime.com/ru/ (дата обращения: 30.04.2019).

9. Арлоу Д., Нейштадт И. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2007. – 624 с.

10. Головатый А., Каплан-Мосс Дж. Django. Подробное руководство, 2-е изд. Пер. с англ. – СПб.: Символ-Плюс, 2013. – 560 с.

11. Документация по Django Framework. [Электронный ресурс] URL: https://docs.djangoproject.com/en/2.2/ (дата обращения: 29.04.2019).

12. Исследование уровня информационной безопасности – 2018. [Электронный ресурс] URL: https://searchinform.ru/research-2018/ (дата обращения: 23.05.2019).

13. Применение ТРИЗ-эволюционного подхода к исследованию объектно-ориентированных языков программирования / В. Д. Бердоносов, А. А. Животова; Комсом.-на-Амуре гос. техн. ун-т. – Комсомольск-на-Амуре: КнАГТУ, 2014. – 165 с.

14. Система контроля и учета рабочего времени сотрудников KickIdler. [Электронный ресурс] URL: https://www.kickidler.com/ru/ (дата обращения: 30.04.2019).

15. Система контроля сотрудников Стахановец. [Электронный ресурс] URL: https://stakhanovets.ru/ (дата обращения: 30.04.2019).

16. Слежение за компьютером NeoSpy. [Электронный ресурс] URL: https://ru.neospy.net/ (дата обращения: 30.04.2019).

17. Чан У., Биссекс П., Форсье Д. Django. Разработка веб-приложений на Python. – СПб.: Символ-плюс, 2009. – 456 с.

18. Что такое DLP и как они работают? [Электронный ресурс] URL: https://searchinform.ru/informatsionnaya-bezopasnost/dlp-sistemy/ (дата обращения: 23.05.2019).

19. Юридическая справка. Разрешено ли следить за сотрудниками. [Электронный ресурс] URL: https://www.kickidler.com/ru/legal-help.html (дата обращения: 25.05.2019).