**Оглавление**

[**Оглавление** 4](#_Toc510262922)

[**Определения, обозначения и сокращения** 8](#_Toc510262923)

[**Введение** 10](#_Toc510262938)

[**1 АНАЛИЗ И моделирование бизнес-процессов** 12](#_Toc510262946)

[**1.1 Описание объекта автоматизации** 12](#_Toc510262947)

[**2 ФОРМИРОВАНИЕ ТРЕБОВАНИЙ** 13](#_Toc510262957)

[**2.1 Анализ необходимости автоматизации** 13](#_Toc510262958)

[**Программная архитектуры разрабатываемого компонента** 14](#_Toc510262964)

**Определения, обозначения и сокращения**

*ПО* — программное обеспечение.

*ЯП* — язык программирования.

**Введение**

В области разработки программного обеспечения довольно часто поднимается вопрос производительности разрабатываемых программ. Требование к производительности — одно из важнейших нефункциональных требований для большинства продуктов.

На пути от написания кода программы до исполнения соответствующего ей машинного кода на процессоре компьютера пользователя есть множество факторов, которые так или иначе могут влиять на производительность разрабатываемой программы.

Не углубляясь в конкретные факторы, выделим стадии с этими факторами от написания кода до исполнения машинного кода программы на процессоре:

* написание кода программистом,
* компиляция кода (некоторые стадии могут отсутствовать, либо быть совмещены):
  1. лексический анализ,
  2. синтаксический анализ,
  3. семантический анализ,
  4. оптимизации на абстрактном синтаксическом дереве,
  5. трансляция в промежуточное представление,
  6. машинно-независимые оптимизации,
  7. трансляция в конечное представление (машинный код),
  8. машинно-зависимые оптимизации.
* интерпретация или компиляция «на лету» промежуточного кода виртуальной машиной (при трансляции кода компилятором не в машинный код, а в код некоторой виртуальной машины),
* планирование исполнения машинного кода ядром операционной системы,
* исполнение кода программы на процессоре,
* исполнение кода функций библиотек поддержки времени исполнения, используемых в программе,
* исполнение кода системных вызовов, используемых в программе.

На каждой из перечисленных стадий существует множество факторов, способных в конечном счете повлиять на производительность программы. Под контролем программиста непосредственно целевой программы находится лишь одна группа факторов. За остальные группы факторов ответственны разработчики соответствующих инструментов и вспомогательных программ или их частей: виртуальных машин, библиотек поддержки времени исполнения, ядер операционных систем и непосредственно самих процессоров и других физических компонентов, способных влиять на производительность исполняемой программы.

В данной работе предлагается провести исследование, связанное с анализом влияния на производительсть программ групп факторов в рамках стадий написания кода и его компиляции.

Хочется сразу отметить, что результаты такого анализа могут быть использованы как разработчиками целевых программ, так и разработчиками языка программирования, на котором эти программы составляются.

В качестве языка программирования, программы на котором и компилятор которого будут исследоваться, был выбрал Kotlin, как один из наиболее интересных и быстро развивающихся языков.

Таким образом, целью данной магистерской диссертации является разработка набора инструментов для анализа исходного кода программ на ЯП Kotlin и выявления потенциальных проблем производительности в них, а также проведение исследования по данной теме с использованием разработанного набора инструментов.

Практическая значимость работы заключается в

В качестве темы данной выпускной квалификационной работы и была предложена разработка компонента для сопровождения такого программного продукта. Сопровождение будет заключаться в использовании разрабатываемого компонента для управления внутренней информационной базой этого программного продукта.

Таким образом, целью данной выпускной квалификационной работы является проектирование и разработка приложения, позволяющего автоматизировать работу с информационной базой сканера безопасности.

На данный момент работа с информационной базой сканера безопасности осуществляется вручную, процесс занимает длительное время.

Практическая значимость работы заключается в получении по результатам разработки инструмента удобного управления информационной базой сканера безопасности, который позволит существенно сэкономить время поиска, просмотра и занесения информации, необходимой работы сканера безопасности.

Функционал разработанного компонента планируется продемонстрировать с помощью снимков экрана, сопроводив их соответствующим подробным описанием.

**1 АНАЛИЗ И моделирование бизнес-процессов**

**1.1 Описание объекта автоматизации**

Компания, для которой будет рассматриваться возможность автоматизации, занимается исследованиями в области информационной безопасности, проведением аудитов в области информационной безопасности и разработкой программных продуктов, связанных с безопасностью корпоративных информационных систем.

Ключевой продукт компании – сканер безопасности ERPScan Security Monitoring Suite. Сканер безопасности сканирует различные системы на предмет уязвимостей, конфликтов прав и других проблем в безопасности, используя специальную информационную базу. Эта база содержит в себе следующую информацию:

* проверки,
* сигнатуры,
* правила идентификации серверов приложений,
* параметры для версий компонентов целевых систем,
* SAP Security Notes.

Данная информация заносится и редактируется сотрудниками различных подразделений компании с различной частотой.

В целом, сама база и работа с ней играют ключевую роль в функционировании сканера безопасности.

**2 ФОРМИРОВАНИЕ ТРЕБОВАНИЙ**

**2.1 Анализ необходимости автоматизации**

На текущий момент работа с информационной базой сканера безопасности осуществляется без использования специального программного обеспечения, адаптированного под предметную область и учитывающего её специфику: занесение и редактирование проверок, сигнатур, SAP Security Notes и правил идентификации серверов приложений, а также указание версий компонентов целевых систем осуществляется путем использования соответствующих функций веб-интерфейса



Рисунок 2.10.1 - Диаграмма вариантов использования средства автоматизации.

* 1. кортеж для версии).

ПРИЛОЖЕНИЕ Б

**Программная архитектуры разрабатываемого компонента**



Рисунок Б.1 – Изображение программной архитектуры в развернутом виде