Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра программного обеспечения информационных технологий

К защите допустить:

Заведующий кафедрой ПОИТ

_____ Н. В. Лапицкая

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему

ПРОГРАММНОЕ СРЕДСТВО WEB ПРИЛОЖЕНИЕ ДЛЯ ПРОДАЖИ КВАРТИР

БГУИР ДП 1-40 01 01 03 *** ПЗ

Студент	Е. М. Новик
Руководитель	Д. В. Горбачев
Консультанты:	
от кафедры ПОИТ	Д. В. Горбачев
по экономической части	Е. В. Анохин
Нормоконтролёр	В. А. Леванцевич

Рецензент

Минск 2017

РЕФЕРАТ

Пояснительная записка 104с., 16 рис., 6 табл., 18 формул и 0 литературный источник.

ЗАПУТЫВАНИЕ, VHDL, ПАРСИНГ, ОБФУСКАЦИЯ, RTL

Предметной областью разработки является сфера защиты интеллектуальной собственности, анализа кодов и обфускации. Объект разработки – приложение для конечного пользователя, предоставляющее функционал по анализу и запутыванию кода.

Целью разработки является создание удобного, простого приложения, пригодного для решения практических задач, возникающих при работе в области моделирования дизайнов на языке VHDL.

При разработке проекта использовалась среда разработки Sublime Text 3 с различными расширениями, такими как, LatexTools, Package Control и т.д. Язык программирования приложения – Ruby.

Результатом разработки стало простое в использовании приложение, которое может быть легко интегрировано в процесс работы, предоставляющее различные возможности по проверке и запутыванию кода. Разработаны диаграмма компонентов, диаграмма потоков данных, а также различные схемы алгоритмов.

Предполагается использование приложения разработчиками цифровых микросхем.

Разработанное приложение является экономически эффективным, оно полностью оправдывает средства, вложенные в его разработку.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

Инициализация – приведение областей памяти в состояние, исходное для последующей обработки или размещения данных.

Программа – данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма.

Программное обеспечение – программы, процедуры, правила и любая соответствующая документация, относящиеся к работе вычислительной системы.

Программирование – практическая деятельность по созданию программ.

Программный модуль – программа или функционально завершенный фрагмент программы, предназначенный для хранения, трансляции, объединения с другими программными модулями и загрузки в оперативную память.

Подпрограмма – программа, являющаяся частью другой программы и удовлетворяющая требованиям языка программирования к структуре программы.

Спецификация программы – формализованное представление требований, предъявляемых к программе, которые должны быть удовлетворены при ее разработке, а также описание задачи, условия и эффекта действия без указания способа его достижения.

Веб-интерфейс – это совокупность средств, при помощи которых пользователь взаимодействует с веб-сайтом или любым другим приложением через браузер.

ООП – объектно-ориентированное программирование

ПС – программное средство

ОС – операционная система

БД – база данных

СУБД – система управления базами данных

JPA – спецификация Java EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных

DAO – data access object

ВВЕДЕНИЕ

При любом общественном устройстве особое место в системе общественных отношений занимает недвижимое имущество, так как с его функционированием связаны жизнь и деятельность людей во всех сферах бизнеса, управления и организации. Именно недвижимость формирует центральное звено всей системы рыночных отношений. Объекты недвижимости – не только важнейший товар, удовлетворяющий разнообразные личные потребности людей, но одновременно и капитал в вещной форме, приносящий доход. Основным видом сделок с недвижимостью является купля-продажа.

Такой бизнес, как купля-продажа недвижимости, несомненно, нуждается в компьютеризации своей деятельности и переходе от работы с бумажными документами к работе с электронными данными. Помимо этого, поиск необходимой и доступной недвижимости очень сложен, если использовать старые методы как поиск объявлений в газетах, прослушивание новости по телевизору и т.д. Для поиска такой информации, потенциальный клиент тратит много своего времени.

Целям оптимизации процесса в данной ситуации может послужить создание web приложения для продажи квартир, с помощью которой риелтор смогут частично (а в будущем полностью) отказаться от ведения записей на бумаге и которая поможет автоматизировать некоторые процессы по покупке недвижимости. А клиенты смогут искать подходящую недвижимость в интернете, используя для этого различные критерии поиска. Более того, человек сможет сам создать объявление для продажи недвижимости, указав всю необходимую информацию для этого.

Актуальность и значимость предлагаемой работы заключена как в теоретическом, так и практическом плане, поскольку операции с недвижимым имуществом стали массовыми и повседневными в предпринимательской деятельности граждан и юридических лиц. В данном дипломном проекте будет разработано ПС для продажи квартир, которое поможет автоматизировать многие процессы для риелторов, а так же предоставит легкий способ для поиска необходимой недвижимости или размещений объявлений для продажи квартир.

1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

В данном разделе будет произведён обзор предметной области задачи, решаемой в рамках дипломного проекта;

1.1 Предметная область

Объектом исследования данной работы является порядок купли-продажи жилых помещений, который регулирует вопрос прав человека на жилое помещение. Под предметом исследования понимаются современные проблемы регулирования продажи жилого помещения.

Следует отметить, что современный рынок жилья существенно отличается от рынка прошлых лет. Круг традиционных участников договора купли-продажи значительно расширился. Теперь обычно в сделку кроме покупателя и продавца включается третья сторона - посредник, в качестве которого выступает либо риэлтерская фирма, либо частный маклер. В сделках на рынке жилья все большую роль играют коммерческие банки, биржи, страховые компании, инвестиционные фонды и другие рыночные институты.

Учитывая высокую стоимость жилой площади, сумма заключаемых сделок оценивается в больших деньгах. Квартирный бизнес является одним из самых выгодных. Посреднические организации - риэлтерские фирмы и отдельные маклеры готовы оказать любые услуги гражданам по распоряжению принадлежащими им квартирами и домами, получая при этом наибольшие дивиденды.

К договору купли-продажи жилья применяются обязательные требования:

– письменная форма в виде одного документа, подписанного сторонами, с государственной регистрацией сделки и нового собственника;
– указание имени и регистрации по месту жительства;
– определенная (однозначная) характеристика предмета сделки;
– данные о возможных правах третьих лиц;
– цена жилого помещения и оплата расходов по договору;
– срок и порядок передачи имущества.
– Покупатели квартир преследуют различные цели:
– улучшение жилищных условий;
– перемена места жительства;
– вложение капитала в недвижимость для последующей продажи и

получения дохода.

Купля и продажа недвижимости включает в себя сложные процессы, оказывающие различные услуги для клиентов. Такие как обмен, продажа, покупка недвижимости и аренде жилья. В данной бизнес сфере исчерпывающие базы данных, содержащие информацию обо всех актуальных предложениях на рынке недвижимости, что позволяет предоставить клиенту информацию о предлагаемом объекте, полностью соответствующем его индивидуальным запросам. Такая сфера, несомненно, нуждается в компьютеризации своей деятельности, поскольку такой объем информации сложно обрабатывать людьми. Программное средство может автоматизировать данный процесс. Например, предоставить сервис для поиска недвижимости по различным критериям. В этом поиске можно использовать следующие критерии: площадь квар-тир, количество комнат, этаж, и другая полезная информация. Это позволяет экономить время потенциальных клиентов. Помимо этого существуют клиенты, которые хотят продать недвижимость.

1.2 Обзор существующих аналогов

1.2.1 Программное средство реалт.бай

Реалт.бай это программное средство, позволяющее размещать объявления для продажи недвижимости. Помимо этого предоставляет сервис для аренды жилья.

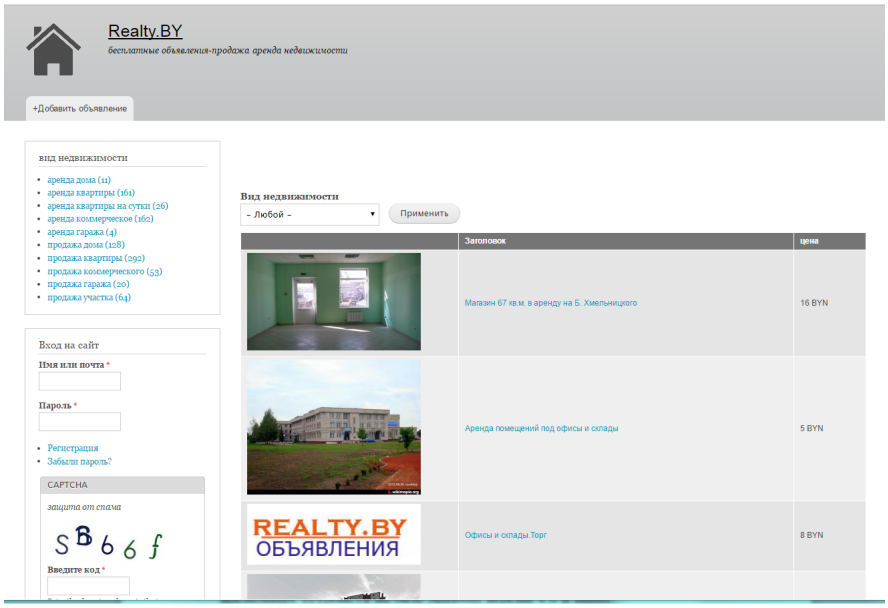

Рисунок 1.1 – Главный страница программного средсва

Реалт.бай это простой способ для размещения объявлений в сети. Веб-интерфейс простой в использовании, все объявления хранятся в базе данных.

Реалт.бай это портал, на котором собрана вся информация о жилой и коммерческой недвижимости в Беларуси. На сайте можно найти информацию по вопросам аренды квартир и комнат в Минске, продаже квартир.

Прежде чем добавить объявление, необходимо пройти процедуру регистрации на сайте. Операция по добавлению нового объявления выглядит следующим образом. Пользователю необходимо добавить описание, тему заголовка, цену, добавить контактные данные. После этого будет размещено объявление.

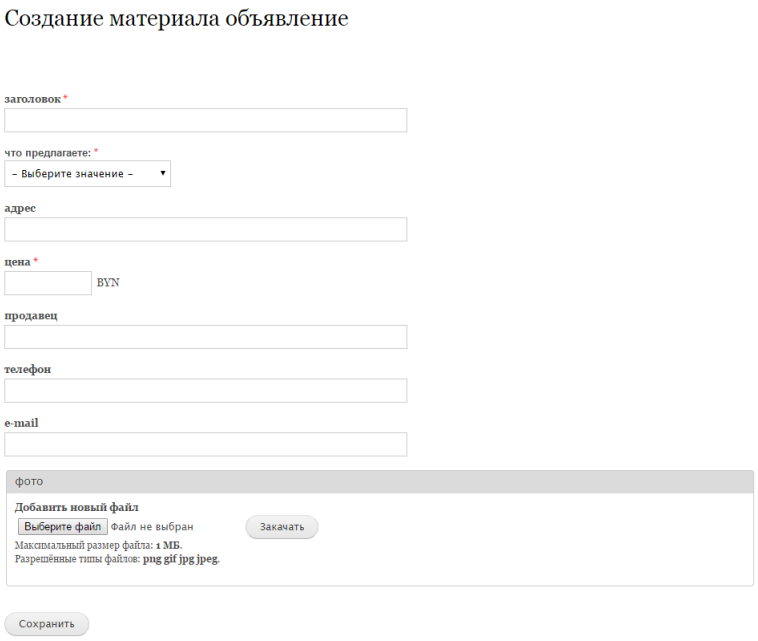

Рисунок 1.2 – Добавление нового объявления

Среди возможностей программного средства реалт.бай можно выделить следующие достоинства:

– понятный и простой интерфейс.

– возможность размещать объявление не только о продажи недвижимости.

– легкая обратная связь между клиентом и риелтором.

Помимо всех плюсов, стоит отметить несколько минусов данного программного средства:

– только один критерий для поиска объявлений.

– нет возможности редактирования информации.

– большинство предложений уже не актуально.

Таким образом, это программное средство подходит в качестве простого приложения, решающее только частично описанные проблемы.

1.2.2 Программное средство квартирант.бай

Квартирант.бай это программное средство, предоставляющее следующий перечень услуг:

– аренда квартир;
– аренда комнат;
– аренда гостинец;
– продажа недвижимости;

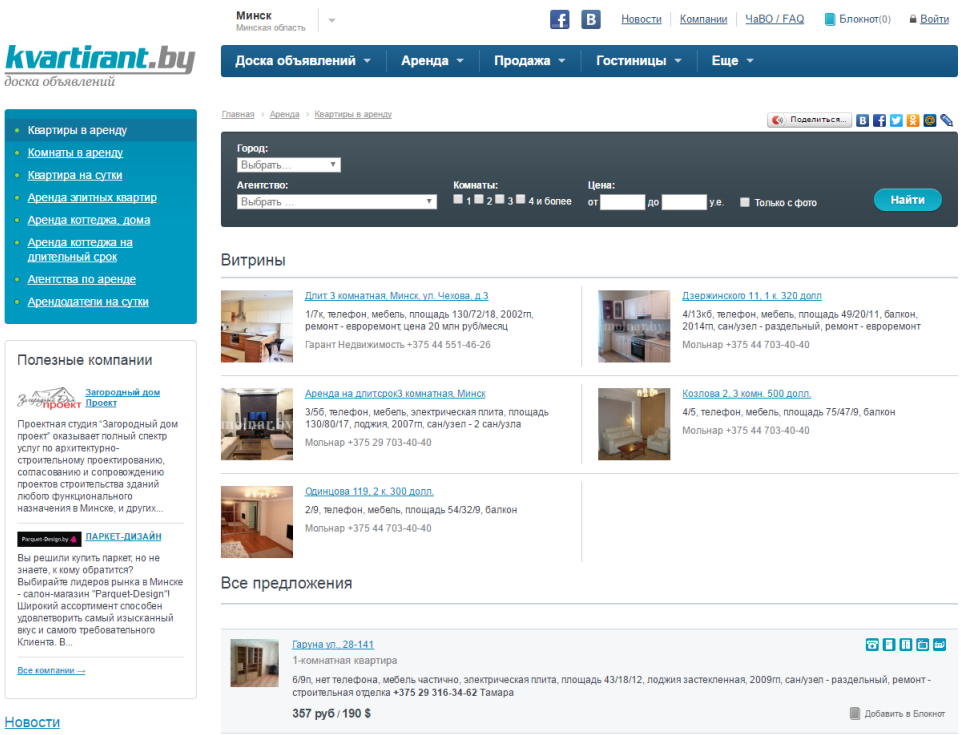

Рисунок 1.3 – Доска объявлений

Проект квартирант.бай стартовал в 2004 году как специализированный сайт по аренде всех видов недвижимости. Доска объявлений портала Квартирант.бай является популярной площадкой для бронирования гостинец, покупки недвижимости, аренды квартир.

Среди возможностей программного средства у можно выделить следующие достоинства:

– понятный интерфейс;
– легкая обратная связь между клиентом и риелтором;

– возможность размещать объявление не только о продажи недвижимости;

Помимо всех плюсов, стоит отметить несколько минусов данного программного средства:

– реклама;
– нет возможности редактирования информации;
– нет возможности оставить отзывы;

Таким образом, это программное средство подходит в качестве приложения, решающе не все описанные проблемы.

1.2.3 Программное средство сильван

Программное средство оказывает услуги в сфере недвижимости. Продажа, покупка, обмен, разъезд, аренда комнат, квартир, коттеджей, дач, земельных участков, коммерческой недвижимости. Приватизация, утверждение перепланировок, вывод в нежилой фонд, консервация и другое.

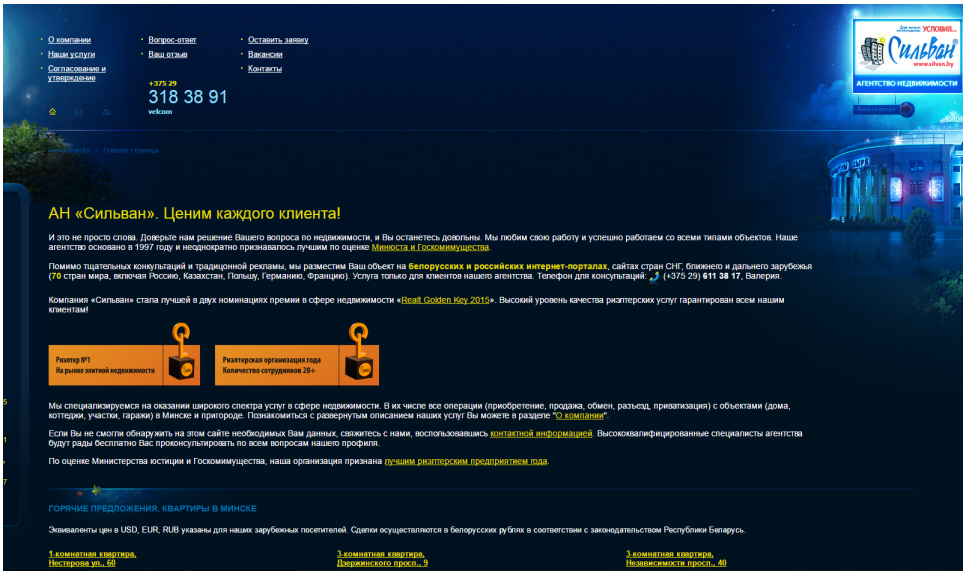

Рисунок 1.4 – Главная форма программного средства

Среди возможностей программного средства у можно выделить следующие достоинства:

– обратная связь между клиентом и риелтором;
– возможность поиска по различным критериям;

Стоит отметить минусы данного программного средства:

– яркий дизайн;
– нет пользователей как таковых
– нет возможности редактирования информации;

– нет возможности оставить отзывы;

– нет возможности добавить объявление;

– все операции делаются в ручную, клиент должен обратиться в офис компании;

Таким образом, данное программное средство не решает множество описанных проблемы. Программное средство представляет из себя частный случай для риэлтерской компании.

1.3 Постановка задачи

В результате выполнения дипломного проекта должно быть разработано программное средство для продажи квартир через веб-интерфейс.

Необходимо выполнить следующие задачи:

– изучить и улучшить знания в web разработке приложения;

– ознакомиться с многопоточными приложениями и особенностями платформы java;

– разработать программное средство для продажи квартир;

– разработать масштабируемое приложение, чтобы была возможность в будущем реализовывать дополнительный функционал;

Программное средство должно иметь два уровня доступа: администратор и пользователь. Пользователь может иметь доступ к своей учетной записи для добавления, поиска, редактирования своих объявлений. Администратор должен иметь возможность настраивать веб-интерфейс и управлять зарегистрированными пользователями.

Должны быть реализованы следующие ключевые функции:

– создание учетной записи пользователя;

– восстановление учетной записи пользователя;

– добавление объявления;

– редактирование объявления;

– удаления объявления;

– добавление фотографий;

– поиск объявления по множеству критериев;

– возможность оставить отзыв;

В процессе дипломного проектирования необходимо разработать приложение, которое будет решать все поставленные задачи и основные функции.

2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ

2.1 Используемые технологии

Выбор технологий является важным предварительным этапом разработки сложных информационных систем. Платформа и язык программирования, на котором будет реализована система, заслуживает большого внимания, так как исследования показали, что выбор языка программирования влияет на производительность труда программистов и качество создаваемого ими кода.

Ниже перечислены некоторые факторы, повлиявшие на выбор технологий:

– разрабатываемое ПО должно иметь возможность запускаться под платформами Windows(7,8,10) и Linux(Ubuntu, Arch Linux, Linux Mint);

– ПО работает в совокупности с другими средствами описания апарутры интегральных схем и должно иметь возможность запускаться в форме скрипта;

– среди различных платформ разработки имеющийся программист лучше всего знаком с разработкой на платформе;

– дальнейшей поддержкой проекта, возможно, будут заниматься разработчики, не принимавшие участие в выпуске первой версии;

– имеющийся разработчик имеет опыт работы с объектоориентированными языками программирования;

Основываясь на опыте работы имеющихся программистов разрабатывать ПО целесообразно с помощью языка Java. Приняв во внимание необходимость обеспечения доступности дальнейшей поддержки ПО, возможно, другой командой программистов, необходимость работы с различными ОС, скриптообразный характер ПО, целесообразно не использовать малоизвестные и сложные языки программирования.

С учетом этого фактора выбор языков программирования сужается до четырех: Python, Ruby и Java. Слабые по сравнению с другими языками механизмы ООП (отсутствие наследования, классов) языка LUA, которые могут быть полезны при разработке ПО, позволяют исключить этот язык из списка кандидатов. Python уступает по удобству использования двум другим кандидатам из нашего списка. Оставшиеся два языка программирования Ruby и Java являются хорошими кандидатами . Тот факт, что Java существует на рынке уже давно, делает этот язык предпочтительным кан-

дидатом.

Таким образом, с учетом вышеперечисленных факторов, целесообразно остановить выбор на следующих технологиях:

– операционные системы: семейство Windows(7,8,10), семейство Linux(Ubuntu, Debian, Arch Linux), Mac os;

– база данных MySql для хранения информации;

– язык программирования Java на котором будет реализована серверная часть программного средства;

– язык программирования Javascript и библиотека react js;

При реализации программного средства встает вопрос, в каком виде должна храниться вся информация и с помощью каких средств её следует обрабатывать. Так как, например, информация о каждом отдельном объявлении представляет собой типичный набор данных (площадь, количество комнат, этаж, год постройки и т. д.), то очевидно, что в этом случае целесообразно хранить их в реляционной базе данных на сервере. Посредством запросов к базе данных пользователь может получать нужные ему сведения, а администратор может добавлять и изменять данные. Выбор конкретной СУБД в качестве сервера баз данных осуществлялся исходя из тех преимуществ, которые она имеет перед другими, а также удобства работы с ней. В данном случае был выбрана клиент-серверная СУБД MySQL

Для реализации поставленной задачи предпочительно использовать на ранних этапах базу данных MySql. В случаи большого количество информации, можно будет слегкость мигрировать на другую СУБД, т.к язык Java использует спецификацию JPA, которая позволяет с легкостью мигрировать на другую СУБД.

Объектно-ориентированный язык программирования Java широко используется для создания серверных приложений. Язык java будет использован для создания высокоуровнего дизайна проложения (иерархия классов и интерфейсов, организация модулей и публичного программного интерфейса), реализации логики приложения, функций и методов [**?**], прототипирования различных идей.

Для реализации клиентской части был выбран язык программирования Javascript. Для быстроты и удобства реализации, будет использована библиотека react.js, которая написана на языке программирования Javascript.

2.1.1 Язык программирования Java

Объектно-ориентированный язык программирования Java широко используется для создания серверных приложений.

Система Java создана на основе простого языка программирования, техника использования которого близка к общепринятой и обучение которому не требует значительных усилий.

Java как язык программирования является объектно-ориентированным с момента основания. Кроме того программист с самого начала обеспечивается набором стандартных библиотек, обеспечивающих функциональность от стандартного ввода/вывода и сетевых протоколов до графических пользовательских интерфейсов. Эти библиотеки легко могут быть расширены.

Несмотря на то, что язык C++ был отвергнут, синтаксис языка Java максимально приближен к синтаксису C++. Это делает язык знакомым широкому кругу программистов. В то же время из языка были удалены многие свойства, которые делают C++ излишне сложным для пользования, не являясь абсолютно необходимыми. В результате язык Java получился более простым и органичным, чем C++.

Надежность и безопасность Java существенно облегчает создание надежного программного обеспечения. Кроме исчерпывающей проверки на этапе компиляции, система предусматривается анализ на этапе выполнения. Сам язык спроектирован так, чтобы вырабатывать у программиста привычку писать "правильно". Модель работы с памятью, в которой исключено использование указателей, делает невозможными целый класс ошибок, характерных для C и C++.

В силу того, что Java предназначен для работы в распределенной среде, безопасность становится чрезвычайно важной проблемой. Требования безопасности определяют многие черты как языка, так и реализации всей системы. Компилятор Java производит байт-коды, т.е. модули приложения имеют архитектурно-независимый формат, который может быть проинтерпретирован на множестве разнообразных платформ. Это уже не исходные тексты, но еще не платформно-зависимые машинные коды.

Схема работы системы и набор байт-кодов виртуальной машины Java таковы, что позволяют достичь высокой производительности на этапе выполнения программы:

анализ кодов на соблюдение правил безопасности производится один раз до запуска кодов на выполнение, в момент выполнения таких проверок уже не нужно, и коды выполняются максимально эффективно ;

работа с базовыми типами максимально эффективна, для операций с ними зарезервированы специальные байт-коды;

методы в классах не обязательно связываются динамически;

автоматический сборщик мусора работает отдельным фоновым потоком, не замедляя основную работу программы, но в то же время обеспечивая своевременный возврат свободной памяти в систему;

стандарт предусматривает возможность написания критических по производительности участков программы в машинных кодах;

Каждая из перечисленных характеристик по отдельности может быть найдена в уже существующих программных пакетах. Новым является соединение их в стройную непротиворечивую систему, которая должна стать всеобщим стандартом.

Java полагается на автоматическое управление памятью со стороны исполняющей среды, предоставляя совсем немного средств для управления жизненным циклом объектов. Не смотря на это, в языке все же присутствуют указатели на функции.

Создатели языка Java не являются противниками привнесения в язык новых идей и возможностей. Каждая новая версия интерпретатора языка привносит различные полезные возможности, которые отвечают требованиям индустрии.

2.1.2 Система управления базами данных MySQL

Выбор конкретной СУБД в качестве сервера баз данных осуществлялся исходя из тех преимуществ, которые она имеет перед другими, а также удобства работы с ней. В данном случае был выбрана клиент-серверная СУБД MySQL. Её архитектура изображена на рисунке ниже.

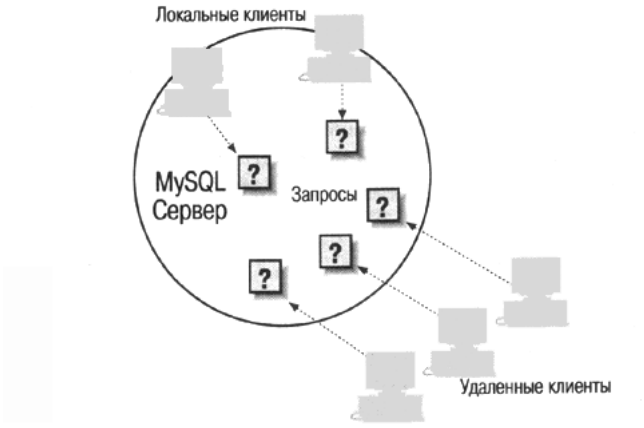

Рисунок 2.1 – Клиент-серверная архитектура MySQL

Клиент-серверная архитектура MySQL Самая подходящая для MySQL сфера применения - это Интернет, благодаря хорошей системе без-

опасности этого пакета, стабильной работе и высокому быстродействию. Для создания скриптов был выбран язык программирования PHP, а MySQL – самая популярная СУБД, которая поддерживается этим языком. В PHP есть множество функций, которые позволяют удобно и эффективно работать с базами данных – и это одна из причин выбора данной СУБД.

Рассмотрим преимущества MySQL:

– Быстродействие. Благодаря внутреннему механизму многопоточности быстродействие MySQL весьма высоко. Для разработчиков MySQL скорость всегда являлась ключевым параметром. Новые возможности добавлялись в пакет MySQL только после того, как их удавалось реализовать без ущерба для производительности. Иногда это означало, что некоторые возможности добавлялись не так быстро, как хотелось бы пользователям, но зато всегда гарантировало быструю работу MySQL.

– Безопасность. Довольно высокий уровень безопасности обеспечивается благодаря базе данных mysql, создающейся при установке пакета и содержащей пять таблиц. При помощи этих таблиц можно описать, какой пользователь из какого домена с какой таблицей может работать и какие команды он может применять. Пароли, хранящиеся в базе данных, можно зашифровать при помощи встроенной в MySQL функции password().

– Лицензия.Раньше лицензирование MySQL было немного запутанным; сейчас эта программа для некоммерческих целей распространяется бесплатно.

– Открытость кода. Благодаря этому программист может сам добавлять в пакет нужные функции, расширяя его функциональность так, как ему требуется. За отдельную плату это могут сделать и сами авторы MySQL.

– Простота использования. Для начала работы с MySQL не требуется сложной процедуры конфигурации. MySQL Server начнёт работать соответствующим образом сразу. По умолчанию выбираются значения, соответствующие минимальному использованию ресурсов диска и памяти. Для получения оптимальной производительности и для специальных условий (например, для проверки входа в систему), конечно же, потребуется дополнительная настройка. Чтобы помочь выполнить такую настройку, предлагаются соответствующие примеры файлов типовой конфигурации.

– Сообщество. Как следствие открытости кода, бесплатности программы, стабильной и надежной ее работы образовалось сообщество людей, которые не просто лояльны к MySQL, но и всячески участвуют как в развитии самого пакета, так и в обучении менее опытных людей работе с ним. Существует огромное количество листов рассылки и конференций, где

можно получить бесплатную помощь в любое время суток.

В настоящее время существуют версии программы для большинства распространенных компьютерных платформ. Это говорит о том, что вам не навязывают определенную операционную систему. Вы сами можете выбрать, с чем работать, например с Linux или Windows, но даже в случае замены ОС вы не потеряете свои данные и вам даже не понадобятся дополнительные инструменты для их переноса. Конечно же, как и любое программное средство, СУБД MySQL не избавлена от некоторых недостатков. Например, можно назвать отсутствие вложенных запросов, что приводит к необходимости находить нужные значения отдельно и подставлять их в другой запрос непосредственно в CGI-сценарии, что, несомненно, сказывается на производительности. Несмотря на это, СУБД MySQL была выбрана как наиболее подходящий сервер баз данных для программного средства.

2.1.3 Язык программирования Javascript

JavaScript — прототипно-ориентированный сценарный язык программирования. Является реализацией языка ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности вебстраницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

Структурно JavaScript можно представить в виде объединения трёх чётко различимых друг от друга частей:

– ядро (ECMAScript),
– объектная модель браузера (Browser Object Model или BOM (en)),
– объектная модель документа (Document Object Model или DOM).

Если рассматривать JavaScript в отличных от браузера окружениях, то объектная модель браузера и объектная модель документа могут не поддерживаться.

Объектную модель документа иногда рассматривают как отдельную от JavaScript сущность, что согласуется с определением DOM как независимого от языка интерфейса документа. В противоположность этому ряд авторов находят BOM и DOM тесно взаимосвязанными.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования

непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания — что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

– объекты, с возможностью интроспекции;
– функции как объекты первого класса;
– автоматическое приведение типов;
– автоматическая сборка мусора;
– анонимные функции;

В языке отсутствуют такие полезные вещи, как:

– JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;

– отсутствует интерфейс программирования приложений по работе с файловой системой, управлению потоками ввода-вывода, базовых типов для бинарных данных;

– стандартные интерфейсы к веб-серверам и базам данных;

– система управления пакетами которая бы отслеживала зависимости и автоматически устанавливала их;

React является библиотекой, написаной на JavaScript с открытым исходным кодом для создания пользовательских интерфейсов. Библиотека была написана разработчиками из Facebook, Instagram и сообществом индивидуальных разработчиков и корпораций.

React позволяет разработчикам создавать крупные веб-приложения, которые используют данные, которые могут меняться со временем, без перезагрузки страницы. Его основная цель - быть быстрым, простым и масштабируемым.

Клиентская часть программного средства, будет разработана с помощью языка Javascript и библиотеки Ract.

2.2 Функциональное моделирование

Для функционального моделирования программного средства для продажи квартир использована диаграмма вариантов использования, являющаяся частью унифицированного языка моделирования (UML). Общий вид обобщенной диаграммы вариантов использования удаленного управления мультимедиа представлен ниже.

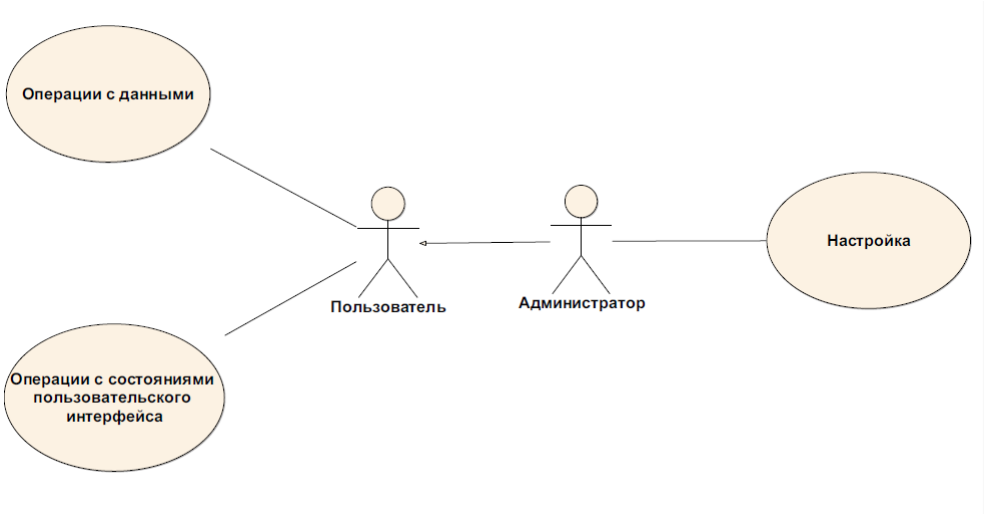

Рисунок 2.2 – Диаграмма вариантов использования

Как видно из диаграммы, для программного средства необходимы два актера:

– пользователь – это любой конечный пользователь из круга лиц, имеющий доступ к удаленному управлению данными на устройствах;

– администратор – это такой пользователь, который кроме всех обычных функций, доступных пользователю, имеет еще специфические функции настройки данных; следует отметить, что администратор наследуются от пользователя.

Среди основных функций пользователя можно выделить:

– операции с данными: все функции, связанные с отображением и управлением мультимедийных данных, дополнительные специфические функции с мультимедиа;

– операции с состояниями пользовательского интерфейса: все функции, предназначенные для управления состояниями пользовательского интерфейса.

У администратора следует выделить функции настройки программного средства, предназначенные настройки всех компонент, связанных с ко-

ректной работой пользователя.

Рассмотрим необходимые функции для работы с объявлениями. диаграмма вариантов использования «Работа с объявлениями» представлена ниже.

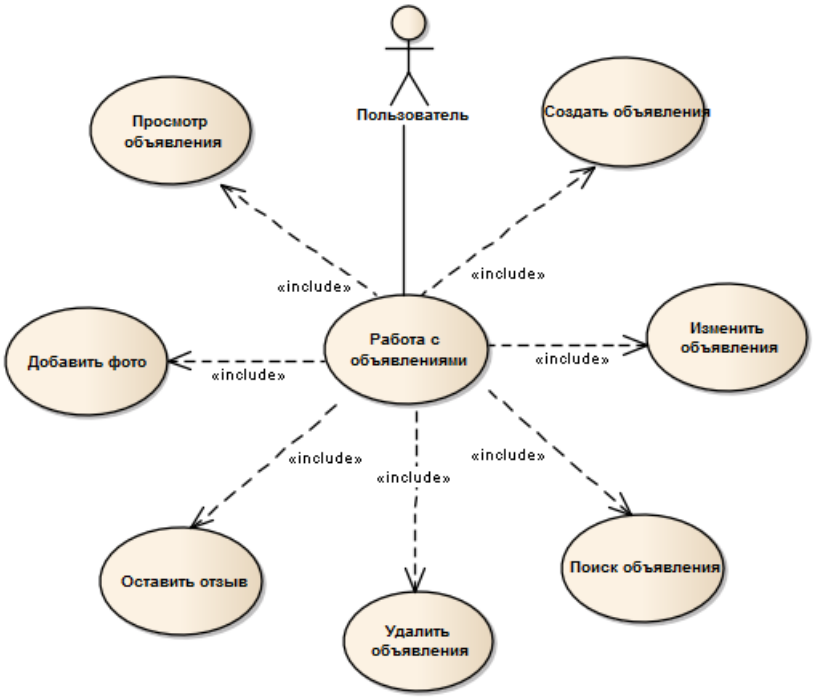

Рисунок 2.3 – диаграмма вариантов использования «Работа с объявлениями»

Как видно из диаграммы, функция «Работа с объявлениями» включает в себя следующие возможности:

– «Создать объявления»: данная функция дает возможность пользователю создавать объявления для возможности выставить свою недвижимость на продажу.

– «Изменить объявление»: функция означает возможность пользователем изменять ранее созданные объявления. Включает возможность редактирования информация, которая была описана при создании объявления.

– «Поиск объявления»: такая функция означает возможность пользователя поиска объявления для покупки или аренды недвижимости. Данная функция включает в себя множество критериев для поиска.

– «Удалить объявления»: функция означает возможность удалить выбранное объявление.

– «Оставить отзыв»: функция дает возможность пользователем добавлять комментарии, задавать дополнительные вопросы на счет недвижимости.

– «Добавить фото»: функция означает возможность пользователю добавить фото о недвижимости.

– «Просмотр объявления»: функция дает возможность пользователям просматривать объявление о недвижимости.

Рассмотрим подробнее функцию «Создать новое объявление». Диаграмма вариантов использования представлена ниже.

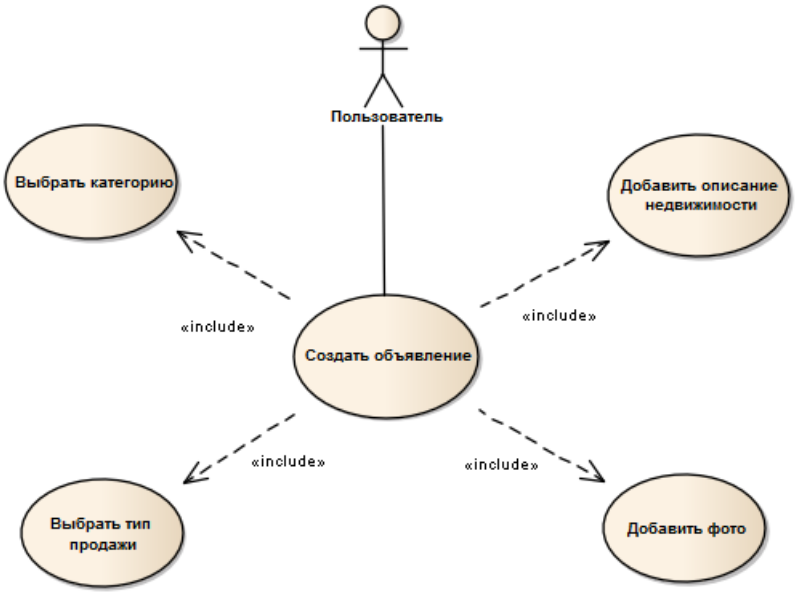

Рисунок 2.4 – Диаграмма вариантов использования «Создать объявление»

Как видно из диаграммы, функция «Создать объявления» включает в себя следующие возможности:

– «Добавить описание недвижимости»: данная функция дает возможность пользователю добавить необходимую информацию о своей недвижимости. Данная информация будет видна другим пользователям.

– «Добавить фото»: функция означает возможность пользователю добавить фотографии о недвижимости. Например пользователь может добавить фотографию планировки квартиры, фотографию гостиной, фотографию кухни и т.д .

– «Выбрать тип продажи»: эта функция дает возможность пользова-

теля выбрать тип продажи недвижимости. Будет это аренда жилья, чистая продажа недвижимости или обмен с доплатой.

– «Выбрать категорию»: эта функция дает возможность пользователя выбрать категорию объявления. В зависимости от категории, будущее объявление будет размещено в нужной секции. Это функция облегчает поиск для других клиентов.

Рассмотрим необходимые функции для роли «Администратор». Диаграмма вариантов использования представлена ниже.

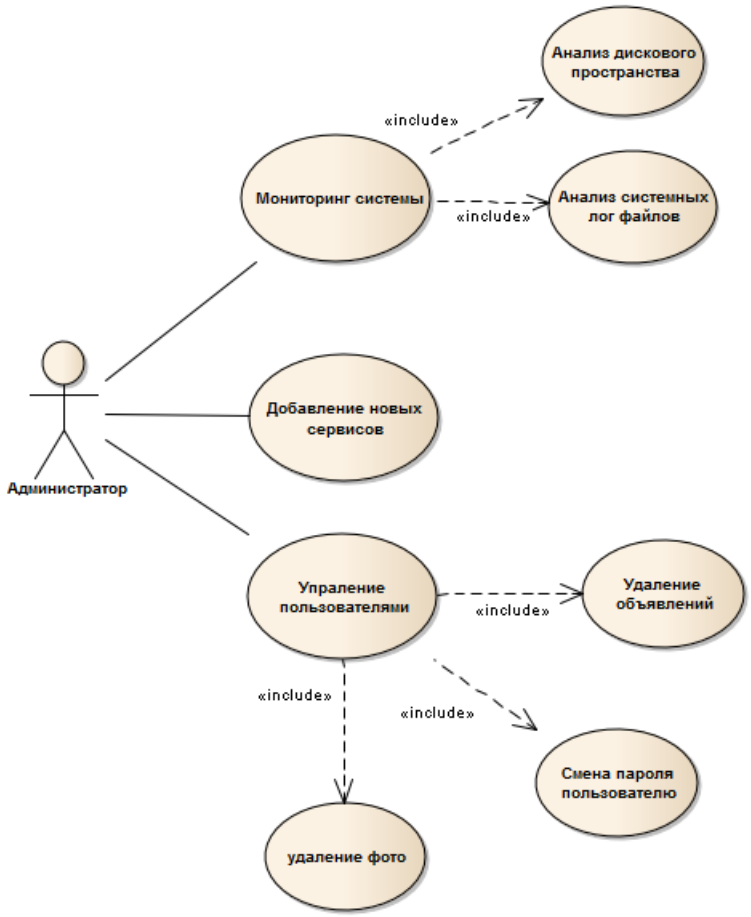

Рисунок 2.5 – Диаграмма вариантов использования для роли «Администратор»

Данная диаграмма показывает, какими функциями должен обладать пользователь, с ролью администратор. Рассмотрим каждую функции в подробном описании:

– «Анализ дискового пространства»: данная функция дает возмож-

ность администратору смотреть свободное место дискового пространства. Данная функция необходима, чтобы периодически чистить устаревшую информацию, которая будет храниться на диске.

– «Анализ системных лог файлов»: функция означает возможность администратору просматривать системные лог файлы на наличие ошибок. В случаи сбоя приложения, пользователю необходима узнать по какой причине произошёл сбой.

– «Добавление новых сервисов»: эта функция дает возможность администратору добавить новый сервис. данная функция предназначена для расширения функционала будущего программного средства.

– «Удаление объявления»: эта функция дает возможность администратору удалить любое объявление. Данная функция предназначена для удаления устаревших и неактуальных объявлений, в случаи, если пользователь сам не удалил объявление.

– «Удаление фото»: эта функция дает возможность администратору удалить любые фотграфии. Данная функция предназначена для удаления устаревших и неактуальных фотографий в случаи, если пользователь сам не удалил объявление.

– «Смена пароля пользователю»: функция дает возможность администратору сменить пароль пользователю. Например, если пользователь не смог восстановить пароль с помощью стандартных методов, он может обратиться в сервисный центр.

Для возможности пользоваться программным средством, необходима учётная запись.

Учётная запись это хранимая в компьютерной системе совокупность данных о пользователе, необходимая для его опознавания (аутентификации) и предоставления доступа к его личным данным и настройкам. Учётная запись, как правило, содержит сведения, необходимые для опознания пользователя при подключении к системе, сведения для авторизации и учёта. Это идентификатор пользователя (login) и его пароль. Пароль, как правило, хранится в зашифрованном или хэшированном виде для обеспечения его безопасности. Для использования учётной записи (другими словами, для входа в систему под чьим-то именем) необходимо использовать логин и пароля. Для регистрации необходимо использовать дополнительное поле email.

Рассмотрим необходимые функции для работы с учетной записью. Данная диаграмма показывает, что пользователь может делать с учетной записью.

Диаграмма вариантов использования представлена ниже.

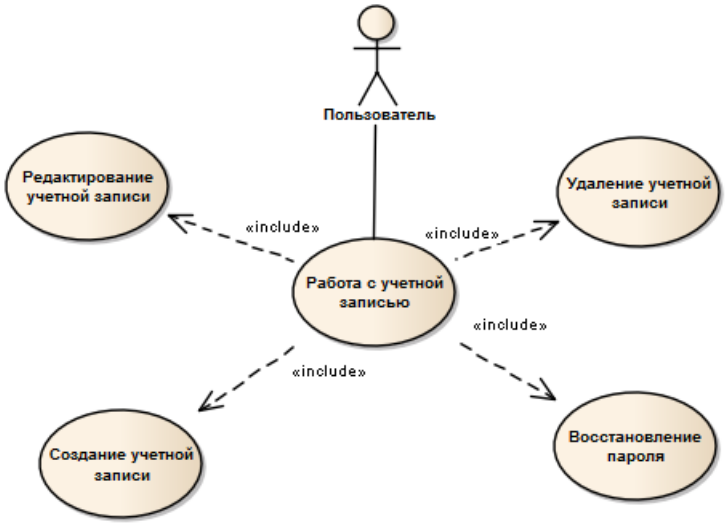

Рисунок 2.6 – Диаграмма вариантов использования «Работа с учетной записью»

Рассмотрим каждую функцию подробнее:

– «Создание учетной записи»: данная функция дает возможность пользователю завести учетную запись, чтобы использовать возможности программного средства.

– «Редактирование учетной записи»: данная функция позволяет пользователю изменить или дополнить информацией свою учетную запись.

– «Удаление учетной записи»: данная функция позволяет пользователю удалить свою учетную запись.

– «Восстановление пароля»: данная функция дает возможность пользователю востановить пароль от учетной записи.

2.3 Разработка спецификации функциональных требований

На основе функциональной модели ПС и поставленных задач необходимо разработать спецификацию функциональных требований к программному средству для продажи квартир.

Спецификация функционального требования «Создание учетной записи пользователя»:

– Функции создания учетной записи должна быть доступна только клиентам;

– При нажатии пользователем на кнопку «sign up» на экране отображается окно, в котором пользователь должен указать логин, пароль, электронную почту;

– После нажатия на кнопку “create” в системе должен появиться новый пользователь. Должно произойти перенаправление на страницу входа в программное средство;

– Если логин или электронная почта уже используется у другого пользователя, ПС должно предложить пользователю ввести другой логин и электронную почту, т.к такие данные уже используются;

Спецификация функционального требования «Восстановление учетной записи пользователя»:

– Функции должна быть доступна только клиентам;

– При нажатии пользователем на кнопку «forgot password» на экране отображается окно, в котором пользователь должен указать свою электронную почту для восстановления пароля;

– После нажатия на кнопку “restore” на почту должны прийти инструкции как восстановить пароль. Должно произойти перенаправление на страницу входа в программное средство;

Спецификация функционального требования «Добавление объявления»:

– Функции должна быть доступна только клиентам;

– При нажатии пользователем на кнопку «forgot password» на экране отображается окно, в котором пользователь должен указать свою электронную почту для восстановления пароля;

– После нажатия на кнопку “restore” на почту должны прийти инструкции как восстановить пароль. Должно произойти перенаправление на страницу входа в программное средство;

Спецификация функционального требования «редактирование объявления»:

– Функции должна быть доступна только клиентам;

– Функция редактирования мультимедиа должна быть доступна, после выбора файла и отмечена галочкой;

– Нажатие пользователем во время редактирования клавиши «Cancel» должно отменить редактирование;

– При нажатии пользователем во время редактирования на клавишу «Enter» редактируемое мультимедиа должно быть сохранено;

– После редактирования выбранного мультимедиа страница должна быть обновлена;

Спецификация функционального требования «удаления объявления»:

– Функции должна быть доступна всем;

– При нажатии пользователем на кнопку удаления должно быть показано окно с подтверждением удаления;

– После удаления выбранного мультимедиа страница должна быть обновлена;

Спецификация функционального требования «добавление фотографий»:

– Функции должна быть доступна только клиентам;

– на странице объявления должна присутствовать кнопка «Add photo». При нажатии на кнопку, появляется диалоговое окно;

– в диалоговом окне должны быть кнопки «choose photo», «add», «cancel»;

– при нажатии кнопки «choose photo» пользователь должен выбрать фото;

– после нажатия кнопки «Add» выбранная фотография появляется на странице объявления;

– при нажатии кнопки «cancel» происходит отмена данной операции;

Спецификация функционального требования «поиск объявления»:

– Функции должна быть доступна только клиентам;

– На главной странице должна присутствовать панель для поиска объявлений;

– На панели должны быть следующие критерии для поиска: ценовой диапазон, город, район, количество комнат, типа продажи, тип категории, кнопка «Search»;

– При нажатии на кнопку «Search» в центре экрана появляются найденные объявления;

Спецификация функционального требования «Добавления отзывы»:

– Функции должна быть доступна только клиентам;

– на странице объявления должна присутствовать кнопка «Add comment». При нажатии на кнопку, появляется диалоговое окно;

– после нажатия кнопки «Add» отзыв появляется на странице объявления;

3 АРХИТЕКТУРА И МОДУЛИ СИСТЕМЫ

Разработанное программное средство представляет из себя клиент-серверное приложение, в котором клиентом выступает браузер, а сервером — веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя. Исходя из выше сказанного, можно сказать, что веб-приложения являются кроссплатформенными сервисами. Серверная часть приложения реализована с помощью языка Java. Клиентская часть реализована с помощью языка Javascript.

Серверная часть состоит из следующих модулей:

– модуль доступа к данным (DAO);
– модуль сервисов(Бизнес логика);
– модуль предоставления данных(Форматирование данных);

Клиентская часть состоит из следующих компонентов:

– модуль действий(actions);
– модуль отображений(components);

Рассмотрим каждый компонент по отдельности.

3.1 Модуль доступа к данным

Данный модуль дает возможность доступа к базе данных. Для реализации данной функциональности использовался шаблон проектирования DAO.

Шаблон проектирования DAO – это объект, который предоставляет абстрактный интерфейс к какому-либо типу базы данных или механизму хранения. Определённые возможности предоставляются независимо от того, какой механизм хранения используется и без необходимости специальным образом соответствовать этому механизму хранения. Этот шаблон проектирования применим ко множеству языков программирования, большинству программного обеспечения, нуждающемуся в хранении информации и к большей части баз данных. На данном уровне нет механизма для управления транзакциями, т.к. транзакции как правило принимают участие в бизнес операциях.

В качестве выбора данного шаблона, послужили следующие факторы:

– DAO инкапсулирует доступ к источнику данных;

– DAO является реализацией слоя объектно-реляционного отображения;

– DAO более ориентирован на источник данных;

Основная цель данного модуля — упростить процесс взаимодействия с БД. Пример реализации шаблона DAO приведены в листинге 3.1:

Листинг 3.1 – Определение для доступа к данным

```
public interface IDAO<Key extends Serializable, E> {

E create(E entity);

E read(Key key);

void update(E entity);

void delete(E entity);

}

public abstract class AbstractDAOImp<Key extends Serializable, E> implements
    IDAO<Key, E> {

@Autowired
private SessionFactory sessionFactory;

private Class<E> clazz;

protected AbstractDAOImp(Class<E> clazz) {
      this.clazz = clazz;
}

protected Session getSession() {
      return sessionFactory.getCurrentSession();
}

@Override
public E create(E entity) {
      Session session = getSession();
      session.persist(entity);
      session.flush();
      return entity;
}

@Override
public E read(Key key) {
      return getSession().find(clazz, key);
}

@Override
public void update(E entity) {
      getSession().update(entity);
```

```
}

@Override
public void delete(E entity) {
	getSession().delete(entity);
}

}

@Repository("UserRepository")
public class UserDAOImp extends AbstractDAOImp<Integer, User> {

protected UserDAOImp() {
	super(User.class);
}

public List<User> readAllUser() {
	return getSession().createQuery("from users", User.class).list();
}

}
```

3.2 Модуль сервисов

На этом уровне реализована бизнес логика приложения, управление транзакциями, сервис для загрузки изображений и другие.

Для начала рассмотрим сервис, который взаимодействует с модулем DAO. Сервис для взаимодейсвтия с DAO приведен в листинге 3.2:

Листинг 3.2 – Использование DAO в сервисах

```
@Service("UserService")
@Transactional
public class UserService {

@Autowired
private UserDAOImp daoImp;

public User addUser(User user) {
	return daoImp.create(user);
}

public void updateUser(User user) {
	daoImp.update(user);
}

public void removeUser(User user) {
	daoImp.delete(user);
}
```

```
public List<User> findAllUser() {
      return daoImp.readAllUser();
}

public User findUserById(int id) {
      return daoImp.read(id);
}

}

@Service("CommentService")
@Transactional
public class CommentService {

@Autowired
private CommentDAOImp daoImp;

public Post addPost(Comment comment){
return daoImp.create(comment);
}

public Post findCommentById(int id){
return daoImp.read(id);
}

public void updateComment(Post post){
daoImp.update(post);
}

public void removeComment(Post post){
daoImp.delete(post);
}

}
```

Сначала может показаться, что модуль сервис повторяет методы класса DAO. Но эти методы не просто повторяют, они оборачивают методы DAO с использованием транзакций. Транзакция это группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена либо целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще, и тогда она не должна произвести никакого эффекта. Можно заметить, что модуль сервисов зависит от модуля DAO.

Рассмотрим сервис для загрузки и сохранения изображений. При разработке приложений была обнаружена проблема с быстрой загрузкой графического контента. Если картинки довольно большие, то необходимо обеспечить эффективную работу с памятью для предотвращения злополучной

ошибки OutOfMemoryError. Так же во время загрузки изображения было необходимо показывать маленькое изображение. Именно данная проблематика и подталкивает обратиться к библиотеке с открытым исходным кодом – Universal Image Loader, целью которой является универсализация решения вышеописанной задачи в виде гибкого и конфигурируемого инструмента.

На данный момент библиотеку можно использовать в тех случаях, когда нужно загрузить и отобразить (и можно еще закэшировать) картинку из интернета или из файловой системы. Классические примеры применения ImageLoader'а – это списки, таблицы, галереи, где необходимо отображать изображения из сети.

Среди возможностей ImageLoader'а можно выделить следующие достоинства:

– асинхронная загрузка и отображение изображений из интернета или с SD-карты;

– возможность кэширования загруженных картинок в памяти и/или на файловой системе устройства;

– возможность отслеживания процесса загрузки посредством «слушателей»;

– эффективная работа с памятью при кэшировании картинок в памяти;

– широкие возможности настройки ImageLoader'а.

К глобальным настройкам ImageLoader'а можно отнести:

– максимальный размер кэшируемых в памяти картинок;

– тайм-аут для установки соединения и загрузки картинки;

– максимальное количество потоков для загрузки изображений, работающих одновременно;

– приоритет потоков по загрузке и отображению картинок;

– программная реализация дискового кэша (можно выбрать одну из готовых реализаций или создать свою собственную);

– программная реализация кэша в памяти (можно выбрать одну из готовых реализаций или создать свою собственную);

– опции загрузки изображения по умолчанию.

Опции загрузки изображения (применяются к каждому отдельному вызову ImageLoader.displayImage(...)) предоставляют возможность указать:

– отображать ли картинку – заглушку в ImageView, пока реальная картинка грузится (если да, то нужно указать эту «заглушку»);

– отображать ли какую-либо картинку в ImageView, если URL картинки был передан пустым (если да, то нужно указать эту картинку);

– кэшировать ли загруженную картинку в памяти;

– кэшировать ли загруженную картинку на файловой системе;

– тип декодирования изображения (максимально быстрый или максимально экономный для памяти).

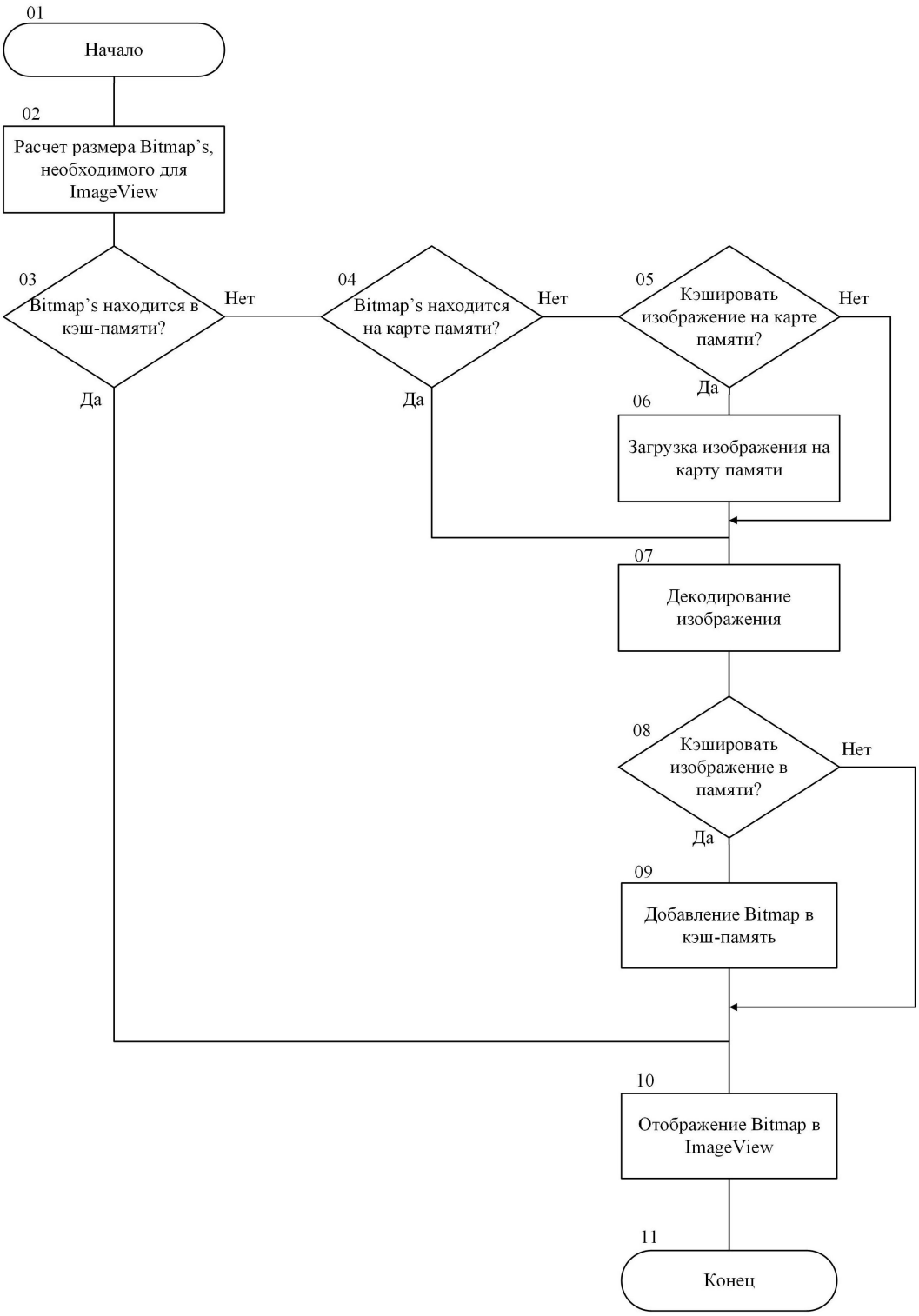

Рисунок 3.1 – Схема алгоритма ImageLoader

Как уже упоминалось, программные реализации дискового кэша и кэша в памяти можно подключить свои. Но, скорее всего, будет достаточ-но готовых решений, которые в большинстве своем представляют собой кэши, ограниченные по какому-либо параметру (размер, количество файлов) и имеющие свою логику самоочищения при превышении лимита (FIFO, самый старый объект, самый большой объект, наиболее редко используемый).

Каждая задача на загрузку и отображение картинки выполняется в отдельном потоке, кроме случаев, если картинка находится в кэше в памяти. Тогда она просто сразу отображается. Существует отдельная очередь потоков, куда попадают задачи, если нужная картинка закэширована на файловой системе.

Если же нужной картинки нет в кэше, то задача-поток попадает в пул пото-ков. Таким образом, быстрому отображению закэшированных картинок ничего не препятствует.

Для управления загрузкой картинок, используется класс ImageLoaderService. Это singletone, поэтому чтобы получить единственный экземпляр класса нужно вызвать метод getInstance(). Перед использованием ImageLoader’a по назна-чению (для отображения картинок), необходимо проинициализировать его конфигурацией.

Самый простой вариант использования ImageLoader’a (с конфигурацией по умолчанию) представлен в листинге 3.3:

Листинг 3.3 – Определение ImageLoaderService

```
@PropertySource(value = { "classpath:imageLoader.properties" })
@Service("ImageLoaderService")
public class ImageLoaderService {

@Autowired
private Environment environment;
private static ImageLoader imageLoader;

public ImageLoader getInstance() {
if(imageLoader == null){
ImageLoader imageLoader = ImageLoader.init(getProperties());
}
return imageLoader;
}


private Properties getProperties() {
Properties properties = environment.loadConfiguration();
return properties;
}

}
```

3.3 Модуль предоставления данных

Следующий за модулем сервисов — модуль предоставляения данных. Для обработки клиентских запросов используются *spring controllers*.

Spring controllers — это классы, которые осуществляют обработку запроса от клиента (браузера). Здесь происходит валидация данных, передача данных в модуль сервисы. Завершающим этапом будет генерация и форматирование ответа в формат *JSON*.

JSON — это текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается. За счёт своей лаконичности по сравнению с XML, формат JSON был более подходящим для сериализации сложных структур. Если говорить о веб-приложениях, формат уместен в задачах обмена данными между браузером и сервером.

Бизнес логика приложения в этом модуле не присутствует, она делегируется модулю сервисов.

В качестве примера рассмотрим controller, который отвечает за регистрацию и входа пользователя в систему. Класс LoginController представлен в листинге 3.4:

Листинг 3.4 – Определение LoginController

```
@Controller
@RequestMapping(value="api/user"
public class LoginController{

@Autowired
private LoginService loginService;

@Autowired
private UserService userService;

@RequestMapping(value="/signin",method=RequestMethod.GET)
public String signIn(HttpServletRequest request, HttpServletResponse response)
    {
String username = request.getParameter("login");
String password = request.getParameter("password");

boolean isValidUser = loginService.isValidUser(username, password);
User user = null;
if(isValidUser){
user = userService.findUserByLogin(username, password);
}

return ObjectMapper.toJSON(user);
}
```

```
@RequestMapping(value="/signup",method=RequestMethod.POST)
public HttpServletResponse signUp(HttpServletRequest request,
   HttpServletResponse response) {
String username = request.getParameter("login");
String password = request.getParameter("password");

boolean isValidUser = loginService.isValidUser(username, password);
if(isValidUser){
userService.createUser(username, password)
response.setAttribute("signUp", "Success operation. You will be redirected to
   sign in page!");
}else{
response.setAttribute("signUp", "Invalid operation. Please, specify correct
   usernamge, email!");
}
return response;
}

}
```

Метод signIn нужен для входа в программное средство. Данный метод включает в себя запрос от клиента и будущий ответ. Для начала входа систему, необходимо получить имя пользователя и его пароль. Если такой пользователь существует, то система загружает пользователя и формирует ответ на клиент. Строка *ObjectMapper.toJSON* преобразует джава объект в формат JSON. Этот ответ будет обработан клиентом.

Метод signUp нужен для регистрации пользователя в программном средстве. Данный метод включает в себя запрос от клиента и будущий ответ. Для регистрации в программном средстве, необходимо получить имя пользователя и пароль, который прислал клиент в запросе. Далее проверяется, если такой пользователь не существует в системе, то пользователь будет создан, иначе происходит уведомление клиента о некорректности передаваемых данных.

3.4 Модуль действий

Рассмотрим модуль действий. Модуль находиться на клиентской части и отвечает за взаимодействие со слоем предоставления данных. Данный модуль общается посредством *http* и *ajax*.

HTTP — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам). Http метод представляет собой последовательность из любых символов, кроме управляющих и разделителей, и опре-

деляет операцию, которую нужно осуществить с указанным ресурсом.

AJAX — подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее. AJAX — не самостоятельная технология, а концепция использования нескольких смежных технологий. AJAX базируется на двух основных принципах:

– использование технологии динамического обращения к серверу «на лету», без перезагрузки всей страницы полностью, например с использованием XMLHttpRequest (основной объект);

– использование DHTML для динамического изменения содержания страницы;

В качестве формата для обмена данных между клиентом и сервером используется JSON.

Первый пример демонструет вход в систему. LoginAction представлен в листинге 3.5:

Листинг 3.5 – Определение LoginAction

```
var LoginAction = {

signIn: function(nickname, password, callback){

      $.ajaxPost("/api/user/signIn", {"nickname": nickname, "password":
         password}, function (data) {
            callback(data);
      });

}

}
```

Рассмотри ситуацию, когда пользователь хочет оставить комментарий на недвижимость. Пользователь может добавить комментарий, удалить комментарий, редактировать комментарий, читать предыдущие комментарии. В рамках этих условий, был разработан модуль CommentsActions, который выполняет все вышеперечисленные действия. CommentsActions представлен в листинге 3.6:

Листинг 3.6 – Определение CommentsActions

```
var CommentsActions = {

addComment: function(text, callback){
```

```
      $.ajaxPost("/api/comments/addComment/", function (data) {
            if (callback) {
                  callback(data);
            }
      });
},

removeComment: function(id, callback){
      $.ajaxPost("/api/comments/removeComment/", {"id":id}, function (data) {
            if (callback) {
                  callback(data);
            }
      });
},

editComment: function(id, text, callback){
      $.ajaxPost("/api/comments/editComment/", {"id":id, "text":text},
         function (data) {
            if (callback) {
                  callback(data);
            }
      });
},

loadAllComments:function(id, callback){
      $.ajaxPost("/api/comments/loadAll/", {"id":id}, function (data) {
            if (callback) {
            callback(data);
            }
      });
}

}
```

Подобным обазом работают и другие функции. Сначала указываться url для запроса на сервер. Далее перечесляются обязательные параметры для сервиса. Следующим параметрам идет функцию в которую придет ответа от севера. Если ответ удволетворяет условию, будет вызвана *callback функция*.

Callback функция или функция обратного вызова в программировании — передача исполняемого кода в качестве одного из параметров другого кода. Обратный вызов позволяет в функции исполнять код, который задаётся в аргументах при её вызове.

В момент, когда ответит ajax запрос, будет вызва функция обратного вызова. Обработка ответа будет представлена в месте, где используется модуль действия.

3.5 Модуль отображения

Рассмотрим модуль отображений. Модуль находиться на клиентской части и отвечает за взаимодействие со слоем действий. Данный модуль непосредственно отображает информацию для конечного пользователя. Для реализации данного модуля, была выбрана библиотека *React*.

React — уровень представления данных. React дает язык шаблонов и некоторые callback-функции для отрисовки HTML. Весь результат работы React — это готовый HTML. Компоненты react занимаются тем, что хранят свое внутреннее состояние в памяти (например: какая закладка выбрана), но в итоге отображается html.

Библиотека была выбрана из-за следующих плюсов:

– Всегда можно сказать, как компонент будет отрисован, глядя на исходный код;

– Связывание JavaScript и HTML в JSX делает компоненты простыми для понимания;

– Можно генерировать html на сервере;

Рассмотрим форму для регистрации, реализованую с помощью библиотеки React. Код представлен в листинге 3.7:

Листинг 3.7 – Форма регистрации LoginAction

```
import React, { PropTypes } from 'react';
import { Link } from 'react-router';
import { Card, CardText } from 'material-ui/Card';
import RaisedButton from 'material-ui/RaisedButton';
import TextField from 'material-ui/TextField';


const LoginForm = ({
onSubmit,
onChange,
errors,
user
}) => (
<Card className="container">
<form action="/" onSubmit={onSubmit}>
<h2 className="card-heading">Login</h2>

{errors.summary && <p className="error-message">{errors.summary}</p>}

<div className="field-line">
<TextField
floatingLabelText="Email"
name="email"
errorText={errors.email}
```

```
onChange={onChange}
value={user.email}
/>
</div>

<div className="field-line">
<TextField
floatingLabelText="Password"
type="password"
name="password"
onChange={onChange}
errorText={errors.password}
value={user.password}
/>
</div>

<div className="button-line">
<RaisedButton type="submit" label="Log in" primary />
</div>

<CardText>Don't have an account? <Link to={'/signup'}>Create one</Link>.</
   CardText>
</form>
</Card>
);

LoginForm.propTypes = {
onSubmit: PropTypes.func.isRequired,
onChange: PropTypes.func.isRequired,
errors: PropTypes.object.isRequired,
user: PropTypes.object.isRequired
};

export default LoginForm;

import React, { PropTypes } from 'react';
import { Link, IndexLink } from 'react-router';


const Base = ({ children }) => (
<div>
<div className="top-bar">
<div className="top-bar-left">
<IndexLink to="/">React App</IndexLink>
</div>

<div className="top-bar-right">
<Link to="/login">Log in</Link>
<Link to="/signup">Sign up</Link>
</div>

</div>
```

```
{children}

</div>
);

Base.propTypes = {
children: PropTypes.object.isRequired
};

export default Base;

import Base from ’./components/Base.jsx’;
import HomePage from ’./components/HomePage.jsx’;
import LoginPage from ’./containers/LoginPage.jsx’;
import SignUpPage from ’./containers/SignUpPage.jsx’;


const routes = {
// base component (wrapper for the whole application).
component: Base,
childRoutes: [

{
path: ’/’,
component: HomePage
},

{
path: ’/login’,
component: LoginPage
},

{
path: ’/signup’,
component: SignUpPage
}

]
};

export default routes;
```

В React имеет компоненты, состояние, рендер. Компонент включает в себя html и функции для управления элементами на странице. Рендер предоставляет конечный вариант HTML браузеру (то, что видит пользователь). В некоторых компонентах нужно сохранять внутреннее состояние, которое используется во время визуализации. Например, флажок должен помнить, что он был выбран. Для этого используется функция состояние.

В завершении, рассмотрим диаграмму компонентов, отражающая работу всего приложения. Диаграмма представлена на рисунке 3.2

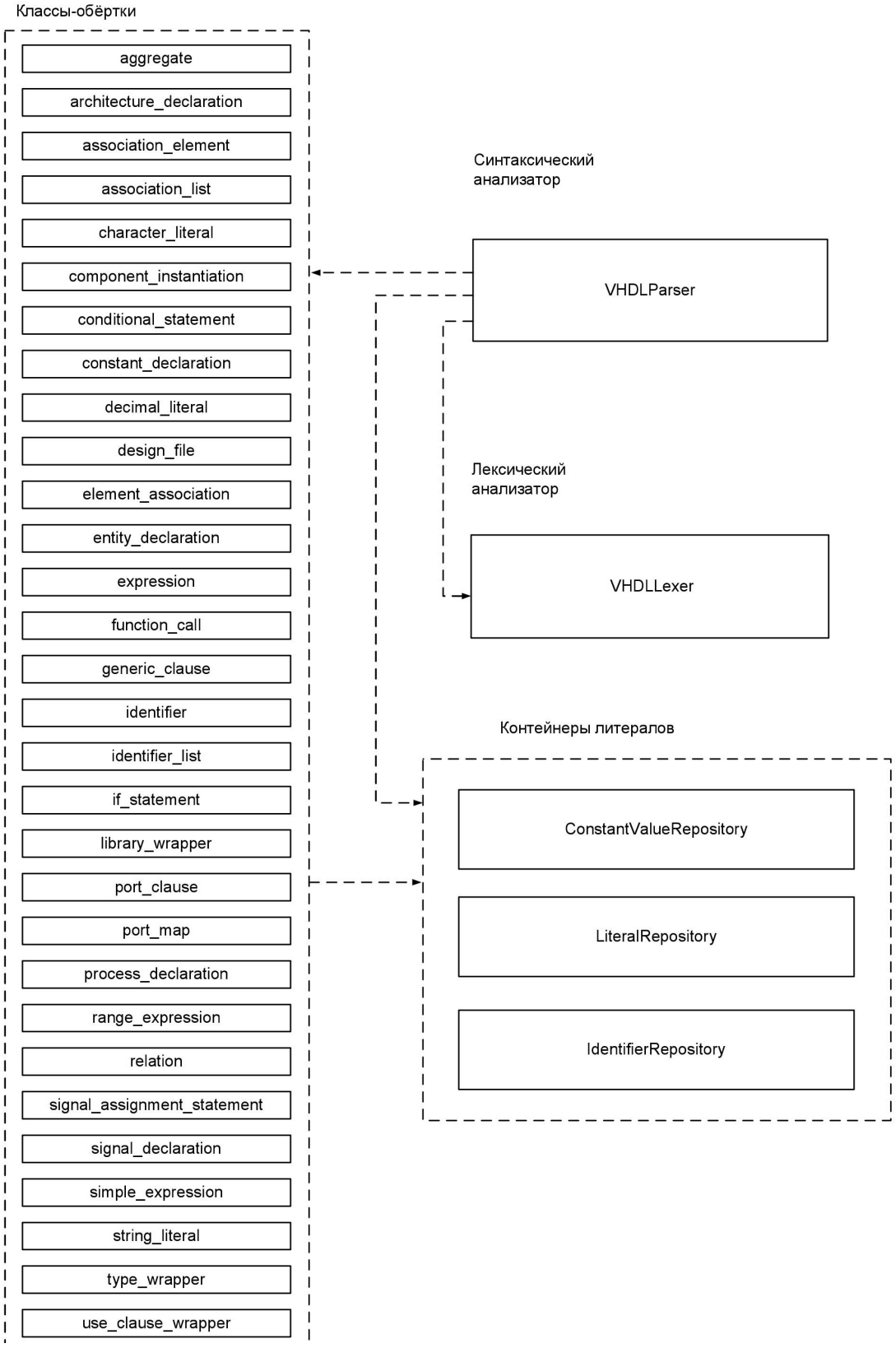

Рисунок 3.2 – Диаграмма компонентов программного средства

4 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

Для оценки правильности работы программного средства было проведено тестирование. Тест-кейсы для функционального требования «Взаимодействие с пользователем» представлены в таблице 4.1

Таблица 4.1 – Тестирование взаимодействия с пользователем

Название тест-кейса и его описание	Ожидаемый результат	Полученный результат
1	2	3
Создание учетной записи пользователя а) Ввести имя, почту, пароль. б) нажать кнопку sign up	а) Отображается сообщение об успешном создании пользователя б) Происходит перенаправление на страницу входа в приложение	Пройден
Вход в приложение используя ранее созданную учетную запись пользователя а) Ввести почту и пароль б) Нажать кнопку sign in	а) Происходит перенаправление на страницу с объявлениями б) Отображается список существующих объявлений	Пройден
Востановление учетной записи пользователя а) Нажать на кнопку forgot password б) Ввести почту при создании пользователя в) Нажать кнопку restore password	а) На указанную почту приходит письмо с новом поролем б) Пользователь успешно входит в систему используя новый пароль в) Отображается список существующих объявлений	Пройден

Продолжение таблицы 4.1

1	2	3
Добавления объявления а) Нажать кнопку create note б) Заполнить обязательные поля в) Нажать кнопку add	а) На вкладке My note отображается новое объявление б) Новое объявление должно быть доступно для поиска	Пройден
Изменить существующее объявление а) Выбрать объявление б) Нажать кнопку Edit в) Изменить несколько полей г) Нажать кнопку Save	а) Измененные данные сохранились в объявлении	Пройден
Удалить существующее объявление а) Выбрать нужное объявление б) Нажать кнопку delete	а) Объявление пропадает со вкладки My notes б) Объявление становиться недоступным для поиска	Пройден
Добавить фотографию в объявление а) Выбрать объявление б) Нажать кнопку Add Picture в) Выбрать изображение на компьютере г) Нажать кнопку Upload	а) Новая фотография добавилась к выброному объявлению	Пройден

Продолжение таблицы 4.1

1	2	3
Поиск объявлений а) Открыть страницу поиска б) Ввести необходимые критерии в) Нажать кнопку Search Notes	а) Найденные объявления отображаются в таблице, в центре экрана б) Найденные обявлнения можно просматривать в новой вкладе	Пройден
Оставить отзыв на объявление а) Выбрать необходимое объявление б) Ввести отзыв в поле Comment в) Нажать кнопку Add Comment	а) Написанный отзыв отображается снизу, под основной информацией объявления б) Отображается значок для удаления коментария	Пройден
Редактирование профиля а) Перейти в раздел Settings б) Надать кнопку Edit в) Изменить необходимые данные г) Нажать кнопку Save	а) Страница обновляется б) Новая информация отображается в профиле пользователя	Пройден

Таким образом, результат тестирования подтверждает, что программное средство web прилложение для продажи квартир функционирует в полном соответствии со спецификацией требований.

5 МЕТОДИКА ИСПОЛЬЗОВАНИЯ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

Программное средство лексической и функциональной обфускации проектных описаний цифровых устройств представляет собой консольное приложение, которое может работать под управлением операционных систем семейства Windows и *nix. Корректная работа приложения гарантируется в ОС, перечисленных в разделе, описывающем анализ предметной области и укрупненную спецификацию требований. Данная методика использования программного сердства составлена с использованием операционной системы Arch Linux и симулятора терминала Sakura. Поскольку приложение является консольным и не содержит промежуточных состояний, в качестве управления используются аргументы, передаваемые при запуске приложения. Запуск приложения без аргументов(или с использованием аргументов -h или -help) и результат его работы представлены на рисунке 5.3.

При запуске приложения в аналитическом режиме проверяется правильность кода, однако не выполняется его обфускация. Анализатор проверяет каждую сущность в дизайне и если она верна, то выводится соответствующее сообщение:

```
~/b/code git:master ❯❯❯ ruby vhdlofb.rb -i examples/johnson.vhd -a
Starting code checking...
Code check for entity 'gen' completed.
Code correct
~/b/code git:master ❯❯❯
```

Рисунок 5.1 – Запуск приложения в аналитическом режиме с корректным входным файлом

Если же исходный код содержит какие-либо ошибки, то пользователь увидит сообщение с ошибкой, которое содержит символ, который привёл к ней:

```
~/b/code git:master ❯❯❯ ruby vhdlofb.rb -i examples/johnson.vhd -a
Starting code checking...

parse error on value "1" (DECIMAL_LITERAL)
~/b/code git:master ❯❯❯
```

Рисунок 5.2 – Запуск приложения в аналитическом режиме с некорректным входным файлом

```
~/b/code git:master ❯❯❯ ruby vhdlofb.rb
      Vhdl obfuscator v.0.3.2
      Eugene Shadura. Belarussian State University of Informatics and Radioelectronics

      Usage:
      ruby vhdlofb.rb -i <path to input file>...
      ruby vhdlofb.rb -i <path to input file> -a
      ruby vhdlofb.rb -i <path to input file> -o <path to output file>
      ruby vhdlofb.rb -i <path to input file> [--lexical-only | --functional-only]
      ruby vhdlofb.rb -h | -help
      ruby vhdlofb.rb -version

      Options:
      -h -help Show this screen.
      -version Show version.
      -i select input file
      -a analyze code correctness without obfuscation
      -o print result to file
      --lexical-only obfuscate literal names only without adding complexity to RTL level
      --functional-only obfuscale RTL level synthesis by adding primitives without changing literal names
```

Рисунок 5.3 – Запуск приложения без аргументов

```
~/b/code git:master ❯❯❯ ruby vhdlofb.rb -i examples/8decoder.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity cvg_1 is
PORT(
src: in std_logic := '1';
q: out std_logic
);
end cvg_1;

architecture gggqwrw of cvg_1 is
signal x: std_logic := '1';
begin
x <= x and src;
q <= x;

end gggqwrw;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity cvg_2 is
PORT(
src: in std_logic := '0';
q: out std_logic
);
end cvg_2;

architecture gggqwrw of cvg_2 is
signal x: std_logic := '0';
begin
x <= x and src;
q <= x;

end gggqwrw;

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity EZVNWMY33Q is generic(J4MW6WYCD: integer:=6 ;WGCVGC: integer:=5 ;ENMJC72: integer:=4 ;G3TP3DM: integer:=3 ;TQW3T4: integer:=1 ;W4F4EWE: integer:=7 ;FN3MJRM: integer:=0 ;KRDF3K: inte
ger:=2);
port(WKACAV66ZF:in std_logic_vector(KRDF3K downto FN3MJRM);KXJEGK2H9:out std_logic_vector(W4F4EWE downto FN3MJRM)
);
end EZVNWMY33Q;
architecture EMPND9Y OF EZVNWMY33Q IS
component cvg_1 is
port (src: in STD_LOGIC := '0';
q: out STD_LOGIC);
end component;
```

Рисунок 5.4 – Запуск приложения без указания выходного файла

6 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПС

Целью дипломного проекта является создание программного средства для обфускации исходных кодов проектных описаний, написанныъ на языке VHDL. Данное программное средство позволяет усложнить чтение злоумышленником как исходных кодов программы(лексическая обфускация), так и результата их синтеза(функциональная обфускация). Данное программное средство обладает рядом достоинств: позволяет проверять правильность исходного кода на предварительном этапе, имеет возможность интеграции с другими инструментами разработки, позволяет проводить только лексическую или только функциональную обфускацию, или обе сразу.

Расчеты выполнены на основе методического пособия [**?**].

6.1 Расчёт сметы затрат и цены программного продукта

Целесообразность создания коммерческого ПО требует проведения предварительной экономической оценки и расчета экономического эффекта. Экономический эффект у разработчика ПО зависит от объёма инвестиций в разработку проекта, цены на готовый программный продукт и количества проданных копий, и проявляется в виде роста чистой прибыли.

Исходные данные для разрабатываемого проекта указаны в таблице 6.1.

На основании сметы затрат и анализа рынка ПО определяется плановая отпускаемая цена. Для составления сметы затрат на создание ПО необходима предварительная оценка трудоемкости ПО и его объёма. Расчет объёма программного продукта (количества строк исходного кода) предполагает определение типа программного обеспечения, всестороннее техническое обоснование функций ПО и определение объёма каждой функций. Согласно классификации типов программного обеспечения [**?** , с. 59, приложение 1], разрабатываемое ПО с наименьшей ошибкой можно классифицировать как ПО методо-ориентированных расчетов.

В данном разделе рассмотрим экономическую эффективность программного средства. Программный комплекс относится ко 2-ой группе сложности. Категория новизны продукта - «В». Для оценки экономической эффективности разработанного программного средства проводится расчет цены и прибыли от продажи одной системы(программы).

Таблица 6.1 – Исходные данные

Наименование	Условное обозначение	Значение
Категория сложности		2
Коэффициент сложности, ед.	$К_с$	1,2
Степень использования при разработке стандартных модулей, ед.	$К_т$	0,9
Коэффициент новизны, ед.	$К_н$	0,7
Годовой эффективный фонд времени, дн.	$Ф_{эф}$	231
Продолжительность рабочего дня, ч.	$Т_ч$	8
Месячная тарифная ставка первого разряда, Br	$Т_{м_1}$	298 000
Коэффициент премирования, ед.	К	1,5
Норматив дополнительной заработной платы, ед.	$Н_д$	20
Норматив отчислений в ФСЗН и обязательное страхование, %	$Н_{сз}$	35
Норматив командировочных расходов, %	$Н_к$	15
Норматив прочих затрат, %	$Н_{пз}$	20
Норматив накладных расходов, %	$Н_{рн}$	100
Прогнозируемый уровень рентабельности, %	$У_{рп}$	35
Норматив НДС, %	$Н_{дс}$	20
Норматив налога на прибыль, %	$Н_п$	18
Норматив расхода материалов, %	$Н_{мз}$	3
Норматив расхода машинного времени, ч.	$Н_{мв}$	15
Цена одного часа машинного времени, Br	$Н_{мв}$	2400
Норматив расходов на сопровождение и адаптацию ПО, %	$Н_{рса}$	30

Общий объём программного продукта определяется исходя из количества и объёма функций, реализованных в программе:

$$V_o = \sum_{i=1}^{n} V_i \,, \tag{6.1}$$

где V_i — объём отдельной функции ПО, LoC;
n — общее число функций.

На стадии технико-экономического обоснования проекта рассчитать точный объём функций невозможно. Вместо вычисления точного объёма

функций применяются приблизительные оценки на основе данных по аналогичным проектам или по нормативам [**?** , с. 61, приложение 2], которые приняты в организации.

Таблица 6.2 – Перечень и объём функций программного модуля

№ функции	Наименование (содержание)	Объём функции, LoC	
		по каталогу (V_i)	уточненный ($V_i^{\text{у}}$)
101	Организация ввода информации	85	60
102	Контроль, предварительная обработка и ввод информации	300	250
103	Анализ входного языка(синтаксический и семантический)	700	690
107	Синтаксический и семантический анализ входного языка и генерация кодов команд	9000	7800
108	Процессор языка	3500	3200
109	Организация ввода/вывода информации с интерактивном режиме	200	150
305	Обработка файлов	300	2500
309	Формирование файла	1000	900
506	Обработка ошибочных и сбойных ситуаций	400	300
507	Обеспечение интерфейса между компонентами	900	890
Итог		16 025	14 490

Перечень и объём функций программного модуля перечислен в таблице 6.2. По приведенным данным уточненный объём некоторых функций изменился, и общий уточненный объём ПО $V_{\text{у}} = 14\,490$ LoC.

6.2 Расчёт нормативной трудоемкости

На основании общего объема ПО определяется нормативная трудоемкость ($\text{Т}_{\text{н}}$) с учетом сложности ПО. Для ПО 2-ой группы сложности, к которой относится разрабатываемый программный продукт, нормативная трудоемкость составит $\text{Т}_{\text{н}} = 306$ чел./дн.

Нормативная трудоемкость служит основой для оценки общей трудоемкости $\mathrm{T_o}$. Используем формулу (6.2) для оценки общей трудоемкости для небольших проектов:

$$\mathrm{T_o} = \mathrm{T_н} \cdot \mathrm{K_c} \cdot \mathrm{K_т} \cdot \mathrm{K_н}\,, \tag{6.2}$$

где $\mathrm{K_c}$ — коэффициент, учитывающий сложность ПО;
$\mathrm{K_т}$ — поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей;
$\mathrm{K_н}$ — коэффициент, учитывающий степень новизны ПО.

Дополнительные затраты труда на разработку ПО учитываются через коэффициент сложности, который вычисляется по формуле

$$\mathrm{K_c} = 1 + \sum_{i=1}^{n} \mathrm{K}_i\,, \tag{6.3}$$

где K_i — коэффициент, соответствующий степени повышения сложности ПО за счет конкретной характеристики;
n — количество учитываемых характеристик.

Наличие двух характеристик сложности позволяет [**?** , с. 66, приложение 4, таблица П.4.2] вычислить коэффициент сложности

$$\mathrm{K_c} = 1 + 0{,}20 = 1{,}2\,. \tag{6.4}$$

Разрабатываемое ПО использует стандартные компоненты. Согласно справочным данным [**?** , с. 68, приложение 4, таблица П.4.5] коэффициент использования стандартных модулей для разрабатываемого приложения $\mathrm{K_т} = 0{,}9$. Разрабатываемое ПО не является новым, существуют аналогичные более зрелые разработки у различных компаний и университетов по всему миру. Влияние степени новизны на трудоемкость создания ПО определяется коэффициентом новизны — $\mathrm{K_н}$. Согласно справочным данным [**?** , с. 67, приложение 4, таблица П.4.4] для разрабатываемого ПО $\mathrm{K_н} = 0{,}7$. Подставив приведенные выше коэффициенты для разрабатываемого ПО в формулу (6.2) получим общую трудоемкость разработки

$$\mathrm{T_o} = 306 \cdot 1{,}2 \cdot 0{,}9 \cdot 0{,}7 \approx 231\,\text{чел./дн.} \tag{6.5}$$

На основе общей трудоемкости и требуемых сроков реализации проекта вычисляется плановое количество исполнителей. Численность испол-

нителей проекта рассчитывается по формуле:

$$Ч_р = \frac{Т_о}{Т_р \cdot Ф_{эф}}, \tag{6.6}$$

где $Т_о$ — общая трудоемкость разработки проекта, чел./дн.;
$Ф_{эф}$ — эффективный фонд времени работы одного работника в течение года, дн.;
$Т_р$ — срок разработки проекта, лет.

Эффективный фонд времени работы одного разработчика вычисляется по формуле

$$Ф_{эф} = Д_г - Д_п - Д_в - Д_о, \tag{6.7}$$

где $Д_г$ — количество дней в году, дн.;
$Д_п$ — количество праздничных дней в году, не совпадающих с выходными днями, дн.;
$Д_в$ — количество выходных дней в году, дн.;
$Д_п$ — количество дней отпуска, дн.

Согласно данным, приведенным в производственном календаре для пятидневной рабочей недели в 2016 году для Беларуси [**?**], фонд рабочего времени составит

$$Ф_{эф} = 366 - 6 - 105 - 24 = 231 \text{ дн.} \tag{6.8}$$

Учитывая срок разработки проекта $Т_р = 6 \text{ мес.} = 0{,}50 \text{ года}$, общую трудоемкость и фонд эффективного времени одного работника, вычисленные ранее, можем рассчитать численность исполнителей проекта

$$Ч_р = \frac{231}{0{,}50 \cdot 231} \approx 2 \text{ рабочих.} \tag{6.9}$$

Вычисленные оценки показывают, что для выполнения запланированного проекта в указанные сроки необходимо два рабочих.

6.3 Расчёт основной заработной платы исполнителей

Информация о работниках перечислена в таблице 6.3.

Таблица 6.3 – Работники, занятые в проекте

Исполнители	Разряд	Тарифный коэффициент	Чел./дн. занятости
Программист I-категории	13	3,04	115
Ведущий программист	15	3,48	116

Месячная тарифная ставка одного работника вычисляется по формуле

$$\text{Т}_\text{ч} = \frac{\text{Т}_{\text{м}_1} \cdot \text{Т}_\text{к}}{\Phi_\text{р}}, \tag{6.10}$$

где $\text{Т}_{\text{м}_1}$ — месячная тарифная ставка 1-го разряда, Br;
$\text{Т}_\text{к}$ — тарифный коэффициент, соответствующий установленному тарифному разряду;
$\Phi_\text{р}$ — среднемесячная норма рабочего времени, час.

Подставив данные из таблицы 6.3 в формулу (6.10), приняв значение тарифной ставки 1-го разряда $\text{Т}_{\text{м}_1} = 298\,000\,\text{Br}$ и среднемесячную норму рабочего времени $\Phi_\text{р} = 160$ часов получаем

$$\text{Т}_\text{ч}^{\text{прогр. I-разр.}} = \frac{298\,000 \cdot 3{,}04}{160} = 5662\,\text{Br/час}; \tag{6.11}$$

$$\text{Т}_\text{ч}^{\text{вед. прогр.}} = \frac{298\,000 \cdot 3{,}48}{160} = 6482\,\text{Br/час}. \tag{6.12}$$

Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле

$$\text{З}_\text{о} = \sum_{i=1}^{n} \text{Т}_\text{ч}^i \cdot \text{Т}_\text{ч} \cdot \Phi_\text{п} \cdot \text{К}, \tag{6.13}$$

где $\text{Т}_\text{ч}^i$ — часовая тарифная ставка i-го исполнителя, Br/час;
$\text{Т}_\text{ч}$ — количество часов работы в день, час;
$\Phi_\text{п}$ — плановый фонд рабочего времени i-го исполнителя, дн.;
К — коэффициент премирования.

Подставив ранее вычисленные значения и данные из таблицы 6.3 в формулу (6.13) и приняв коэффициент премирования $\text{К} = 1{,}5$ получим

$$\text{З}_\text{о} = (5662 \cdot 115 + 6482 \cdot 116) \cdot 8 \cdot 1{,}5 = 16\,836\,504\,\text{Br}. \tag{6.14}$$

Дополнительная заработная плата включает выплаты предусмотрен-

ные законодательством от труде и определяется по нормативу в процентах от основной заработной платы

$$З_д = \frac{З_о \cdot Н_д}{100\%}, \tag{6.15}$$

где $Н_д$ — норматив дополнительной заработной платы, %.

Приняв норматив дополнительной заработной платы $Н_д = 20\%$ и подставив известные данные в формулу (6.15) получим

$$З_д = \frac{16\,836\,504 \cdot 20\%}{100\%} \approx 3\,367\,301\,\text{Br}. \tag{6.16}$$

Расчеты общей суммы расходов и прогнозируемой цены ПО, а также его себестоимости сведены в таблицу 6.4.

Таблица 6.4 – Расчет себестоимости и отпускной цены ПО

Наименование статей	Норматив, %	Методика расчета	Значение, руб.
Отчисления в фонд социальной защиты и обязательного страхования	$Н_{сз} = 35$	$З_{сз} = (З_о + З_д) \cdot Н_{сз}/100$	7 071 332
Материалы и комплектующие	$Н_{мз} = 3$	$М = З_о \cdot Н_{мз}/100$	505 095
Машинное время		$Р_м = Ц_м \cdot V_о/100 \cdot Н_{мв}$ $Н_{мв} = 15$ машино-часов $Ц_м = 2400\,\text{Br}$	5 216 400
Расходы на научные командировки	$Н_к = 15$	$Р_к = З_о \cdot Н_к/100$	2 525 476
Прочие прямые расходы	$Н_{пз} = 20$	$П_з = З_о \cdot Н_{пз}/100$	3 367 301
Накладные расходы	$Н_{рн} = 100$	$Р_н = З_о \cdot Н_{рн}/100$	16 836 504
Общая сумма расходов по смете		$С_р = З_о + З_д + З_{сз} + М + Р_м + Р_к + П_з + Р_н$	55 725 913

Сопровождение и адаптация ПО	$\text{Н}_{\text{рса}} = 30$	$\text{Р}_{\text{са}} = \text{С}_{\text{р}} \cdot \text{Н}_{\text{рса}}/100$	16 717 774
Полная себестоимость ПО		$\text{С}_{\text{п}} = \text{С}_{\text{р}} + \text{Р}_{\text{са}}$	72 443 687
Прогнозируемая прибыль	$\text{У}_{\text{рп}} = 35$	$\text{П}_{\text{с}} = \text{С}_{\text{п}} \cdot \text{У}_{\text{рп}}/100$	25 355 290

Продолжение таблицы 6.4

Наименование статей	Норматив, %	Методика расчета	Значение, руб.
Прогнозируемая цена без налогов		$\text{Ц}_{\text{п}} = \text{С}_{\text{п}} + \text{П}_{\text{с}}$	97 798 977
Отчисления и налоги в местный и республиканский бюджеты	$\text{Н}_{\text{мр}} = 3{,}9$	$\text{О}_{\text{мр}} = \text{Ц}_{\text{п}} \cdot \text{Н}_{\text{мр}}/100 - \text{Н}_{\text{мр}}$	3 968 949
Налог на добавленную стоимость	$\text{Н}_{\text{дс}} = 20$	$\text{НДС} = (\text{Ц}_{\text{п}} + \text{О}_{\text{мр}}) \cdot \text{Н}_{\text{дс}}/100$	20 353 585
Прогнозируемая отпускная цена		$\text{Ц}_{\text{о}} = \text{Ц}_{\text{п}} + \text{О}_{\text{мр}} + \text{НДС}$	122 121 511

6.4 Расчёт экономической эффективности у разработчика

Важная задача при выборе проекта для финансирования это расчет экономической эффективности проектов и выбор наиболее выгодного проекта. Разрабатываемое ПО является заказным, т.е. разрабатывается для одного заказчика на заказ. На основании анализа рыночных условий и договоренности с заказчиком об отпускной цене прогнозируемая рентабельность проекта составит $\text{У}_{\text{рп}} = 35\%$.

Чистую прибыль от реализации проекта можно рассчитать по формуле

$$\text{П}_{\text{ч}} = \text{П}_{\text{с}} \cdot \left(1 - \frac{\text{Н}_{\text{п}}}{100\%}\right), \tag{6.17}$$

где $\text{Н}_{\text{п}}$ — величина налога на прибыль, %.

Приняв значение налога на прибыль $Н_н = 18\%$ и подставив известные данные в формулу (6.17) получаем чистую прибыль

$$П_ч = 25\,355\,290 \cdot \left(1 - \frac{18\%}{100\%}\right) = 20\,791\,338\,\text{Br}. \tag{6.18}$$

Программное обеспечение разрабатывалось для одного заказчика в связи с этим экономическим эффектом разработчика будет являться чистая прибыль от реализации $П_ч$. Рассчитанные данные приведены в таблице 6.5.

Таблица 6.5 – Рассчитанные данные

Наименование	Условное обозначение	Значение
Нормативная трудоемкость, чел./дн.	$Т_н$	306
Общая трудоемкость разработки, чел./дн.	$Т_о$	231
Численность исполнителей, чел.	$Ч_р$	2
Часовая тарифная ставка программиста I-разряда, Br/ч.	$Т_ч^{\text{прогр. I-разр.}}$	5662
Часовая тарифная ставка ведущего программиста, Br/ч.	$Т_ч^{\text{вед. прогр.}}$	6482
Основная заработная плата, Br	$З_о$	16 836 504
Дополнительная заработная плата, Br	$З_д$	3 367 301
Отчисления в фонд социальной защиты, Br	$З_{сз}$	7 071 332
Затраты на материалы, Br	М	505 095
Расходы на машинное время, Br	$Р_м$	5 216 400
Расходы на командировки, Br	$Р_к$	2 525 476
Прочие затраты, Br	$П_з$	3 367 301
Накладные расходы, Br	$Р_н$	16 836 504
Общая сумма расходов по смете, Br	$С_р$	55 725 913
Расходы на сопровождение и адаптацию, Br	$Р_{са}$	16 717 774
Полная себестоимость, Br	$С_п$	72 443 687
Прогнозируемая прибыль, Br	$П_с$	25 355 290
НДС, Br	НДС	20 353 585
Прогнозируемая отпускная цена ПО, Br	$Ц_о$	122 121 511
Чистая прибыль, Br	$П_ч$	20 791 338

6.5 Выводы по технико-экономическому обоснованию

Программное средство лексической и функциональной обфускации проектных описаний цифровых устройств является выгодным программным продуктом. Чистая прибыль от реализации ПС ($П_ч$ 20 791 338 рублей) остается организации-разработчику и представляет собой экономический эффект от создания нового программного средства. В итоге было произведено технико-экономическое обоснование разрабатываемого проекта, составлена смета затрат и рассчитана прогнозируемая прибыль, а также показана экономическая целесообразность разработки.

ЗАКЛЮЧЕНИЕ

В данном дипломном проекте был рассмотрен вопрос лексического и синтаксического анализа языка VHDL, также различные методики запутывания, как исходного кода, так и результата синтеза этого кода. В рамках дипломного проекта была разработана библиотека кода для анализа и обфускации исходных кодов на языке VHDL. В разработанном проекте был использован генератор парсеров YACC для создания анализатора языка, а так же произвольный набор правил, построенный на основе BNF-грамматики языка и обфускатор, представленный наборов классов.

В целом получены хорошие результаты обработки и обфускации на ряде исходных кодов. Время работы приложения является линейным и зависит от объёма входных данных. Результаты работы реализованных в проекте функций замены различных типов литералов в большинстве случаев превосходят по качеству функциональность уже существующих открытых аналогов.

В итоге получилось раскрыть тему дипломного проекта и создать в его рамках программное обеспечение. Но за рамками рассматриваемой темы осталось еще много других алгоритмов синтаксического и лексического анализа, а также различных приёмов обфускации.

В дальнейшем планируется развивать и довести существующее ПО до полноценной библиотеки, способной решать более широкий класс задач, возникающих в области защиты интеллектуальной собственности и запутывания кода.

ПРИЛОЖЕНИЕ А
(обязательное)
Исходный код программного средства

```
class VHDLParser


rule

vhdl : design_file {result = val}

abstract_literal :
  DECIMAL_LITERAL {result = DecimalLiteral.new(val[0]);LiteralRepository.add(
     result);}
  | BASED_LITERAL {result = val[0]}

access_type_definition :
  ACCESS subtype_indication {result = val}

actual_designator :
  expression {result = val[0]}
  | signal_name {result = val[0]}
  | variable_name {result = val[0]}
  | file_name {result = val[0]}
  | OPEN {result = val[0]}

actual_parameter_part :
  parameter_association_list {result = val}

actual_part :
  actual_designator {result = val[0];}
  | function_name '(' actual_designator ')' {result = val;}
  | type_mark '(' actual_designator ')' {result = val;}

adding_operator :
  '+' {result = val}
  | '-' {result = val}
  | '&' {result = val}

aggregate :
  '(' element_association aggregate_loop0 ')' {result = Aggregate.new(val
     [1..2].flatten);}

aggregate_loop0 :
  ',' element_association aggregate_loop0 {result = val}
  | {result = val}

alias_declaration :
  ALIAS alias_designator ':' subtype_indication IS name signature ';' {result
     = val}
  | ALIAS alias_designator IS name signature ';' {result = val}
  | ALIAS alias_designator ':' subtype_indication IS name ';' {result = val}
```

```
  | ALIAS alias_designator IS name ';' {result = val}

alias_designator :
  identifier {result = val}
  | CHARACTER_LITERAL {result = CharacterLiteral.new(val[0]);
     ConstantValueRepository.add(result); }
  | operator_symbol {result = val}

allocator :
  NEW subtype_indication {result = val}
  | NEW qualified_expression {result = val}

architecture_body :
   ARCHITECTURE identifier OF type_name IS architecture_declarative_part BEGIN
       architecture_statement_part END ARCHITECTURE identifier ';' {
      InitializeRepository.add(val[1]); result = ArchitectureDeclaration.new(
      val[1], val[3], val[5], val[7],val[9...val.length-1]);}
  | ARCHITECTURE identifier OF type_name IS architecture_declarative_part
     BEGIN architecture_statement_part END identifier ';' {
     InitializeRepository.add(val[1]); result = ArchitectureDeclaration.new(
     val[1], val[3], val[5], val[7],val[9...val.length-1]); }
  | ARCHITECTURE identifier OF type_name IS architecture_declarative_part
     BEGIN architecture_statement_part END ARCHITECTURE ';' {
     InitializeRepository.add(val[1]);result = ArchitectureDeclaration.new(val
     [1], val[3], val[5], val[7],val[9...val.length-1]);}
  | ARCHITECTURE identifier OF type_name IS architecture_declarative_part
     BEGIN architecture_statement_part END ';' {InitializeRepository.add(val
     [1]);result = ArchitectureDeclaration.new(val[1], val[3], val[5], val[7],
     val[9...val.length-1]);}

architecture_declarative_part :
  architecture_declarative_part_loop0 {result = val[0];}

architecture_declarative_part_loop0 :
  block_declarative_item architecture_declarative_part_loop0 {result = val.
     flatten.compact;}
  | {result = val}

architecture_statement_part :
  architecture_statement_part_loop0 {result = val}

architecture_statement_part_loop0 :
  concurrent_statement architecture_statement_part_loop0 {result = val}
  | {result = val;}

array_type_definition :
  unconstrained_array_definition {result = val}
  | constrained_array_definition {result = val}

assertion :
  ASSERT condition REPORT expression SEVERITY expression {result = val}
  | ASSERT condition SEVERITY expression {result = val}
  | ASSERT condition REPORT expression {result = val}
```

```
  | ASSERT condition {result = val}

assertion_statement :
  LABEL ':' assertion ';' {result = val}
  | assertion ';' {result = val}

association_element :
  formal_part '=>' actual_part {result = AssociationElement.new(val[0], val
      [2]);}
  | actual_part {result = AssociationElement.new(nil, val[0])}

association_list :
  association_element association_list_loop0 {result = AssociationList.new(val
      .flatten)}

association_list_loop0 :
  ',' association_element association_list_loop0 {result = val[1..2];}
  | {result = val;}

attribute_declaration :
  ATTRIBUTE identifier ':' type_mark ';' {result = val}

attribute_designator :
  attribute_simple_name {result = val}

attribute_name :
  prefix signature '\'' attribute_designator '(' expression ')' {result = val;
       }
  | prefix '\'' attribute_designator '(' expression ')' {result = val}
  | prefix signature '\'' attribute_designator {result = val}
  | prefix '\'' attribute_designator {result = val}

attribute_simple_name :
  label {result = val[0]; InitializeRepository.add(result)}

attribute_specification :
  ATTRIBUTE attribute_designator OF entity_specification IS expression ';' {
      result = val}

base_unit_declaration :
  identifier ';' {result = val}

basic_character :
  basic_graphic_character {result = val}
  | format_effector {result = val}

basic_graphic_character :
  upper_case_letter {result = val}
  | digit {result = val}
  | special_character {result = val}
  | space_character {result = val}

binding_indication :
```

```
  USE entity_aspect generic_map_aspect port_map_aspect {result = val}
  | generic_map_aspect port_map_aspect {result = val}
  | USE entity_aspect port_map_aspect {result = val}
  | port_map_aspect {result = val}
  | USE entity_aspect generic_map_aspect {result = val}
  | generic_map_aspect {result = val}
  | USE entity_aspect {result = val}
  | {result = val}

block_configuration :
  FOR block_specification block_configuration_loop0 block_configuration_loop1
      END FOR ';' {result = val}

block_configuration_loop0 :
  use_clause block_configuration_loop0 {result = val}
  | {result = val}

block_configuration_loop1 :
  configuration_item block_configuration_loop1 {result = val}
  | {result = val}

block_declarative_item :
  subprogram_declaration {result = val}
  | subprogram_body {result = val}
  | type_declaration {result = val}
  | subtype_declaration {result = val}
  | constant_declaration {result = val[0]; }
  | signal_declaration {result = val[0]; result.type.append_semicolon = true}
  | shared_variable_declaration {result = val}
  | file_declaration {result = val}
  | alias_declaration {result = val}
  | component_declaration {result = val}
  | attribute_declaration {result = val}
  | attribute_specification {result = val}
  | configuration_specification {result = val}
  | disconnection_specification {result = val}
  | use_clause {result = val}
  | group_template_declaration {result = val}
  | group_declaration {result = val}

block_declarative_part :
  block_declarative_part_loop0 {result = val}

block_declarative_part_loop0 :
  block_declarative_item block_declarative_part_loop0 {result = val}
  | {result = val}

block_header :
  generic_clause generic_map_aspect ';' port_clause port_map_aspect ';' {
      result = val}
  | generic_clause port_clause port_map_aspect ';' {result = val}
  | port_clause port_map_aspect ';' {result = val}
  | generic_clause generic_map_aspect ';' port_clause {result = val}
```

```
  | generic_clause port_clause {result = val}
  | port_clause {result = val}
  | generic_clause generic_map_aspect ';' {result = val}
  | generic_clause {result = val}
  | {result = val}

block_specification :
  architecture_name {result = val}
  | block_statement_label {result = val}
  | generate_statement_label '(' index_specification ')' {result = val}
  | generate_statement_label {result = val}

block_statement :
  block_label ':' BLOCK '(' guard_expression ')' IS block_header
     block_declarative_part BEGIN block_statement_part END BLOCK block_label '
     ;' {result = val}
  | block_label ':' BLOCK IS block_header block_declarative_part BEGIN
     block_statement_part END BLOCK block_label ';' {result = val}
  | block_label ':' BLOCK '(' guard_expression ')' block_header
     block_declarative_part BEGIN block_statement_part END BLOCK block_label '
     ;' {result = val}
  | block_label ':' BLOCK block_header block_declarative_part BEGIN
     block_statement_part END BLOCK block_label ';' {result = val}
  | block_label ':' BLOCK '(' guard_expression ')' IS block_header
     block_declarative_part BEGIN block_statement_part END BLOCK ';' {result =
      val}
  | block_label ':' BLOCK IS block_header block_declarative_part BEGIN
     block_statement_part END BLOCK ';' {result = val}
  | block_label ':' BLOCK '(' guard_expression ')' block_header
     block_declarative_part BEGIN block_statement_part END BLOCK ';' {result =
      val}
  | block_label ':' BLOCK block_header block_declarative_part BEGIN
     block_statement_part END BLOCK ';' {result = val}

block_statement_part :
  block_statement_part_loop0 {result = val}

block_statement_part_loop0 :
  concurrent_statement block_statement_part_loop0 {result = val}
  | {result = val}

case_statement :
  case_label ':' CASE expression IS case_statement_alternative
     case_statement_loop0 END CASE case_label ';' {result = val;}
  | CASE expression IS case_statement_alternative case_statement_loop0 END
     CASE case_label ';' {result = val;}
  | case_label ':' CASE expression IS case_statement_alternative
     case_statement_loop0 END CASE ';' {result = val;}
  | CASE expression IS case_statement_alternative case_statement_loop0 END
     CASE ';' {result = val.join(' ');}

case_statement_alternative :
  WHEN choices '=>' sequence_of_statements {result = val;}
```

```
case_statement_loop0 :
  case_statement_alternative case_statement_loop0 {result = val}
  | {result = val}

choice :
  simple_expression {result = val}
  | discrete_range {result = val}
  | element_simple_name {result = val}
  | OTHERS {result = val}

choices :
  choice choices_loop0 {result = val[0]}

choices_loop0 :
  '|' choice choices_loop0 {result = val}
  | {result = val}

component_configuration :
  FOR component_specification binding_indication ';' block_configuration END
     FOR ';' {result = val}
  | FOR component_specification block_configuration END FOR ';' {result = val}
  | FOR component_specification binding_indication ';' END FOR ';' {result =
     val}
  | FOR component_specification END FOR ';' {result = val}

component_declaration :
  COMPONENT identifier IS local_generic_clause formal_port_clause END
     COMPONENT component_simple_name ';' {result = val}
  | COMPONENT identifier local_generic_clause formal_port_clause END COMPONENT
      component_simple_name ';' {result = val}
  | COMPONENT identifier IS formal_port_clause END COMPONENT
     component_simple_name ';' {result = val}
  | COMPONENT identifier formal_port_clause END COMPONENT
     component_simple_name ';' {result = val}
  | COMPONENT identifier IS local_generic_clause END COMPONENT
     component_simple_name ';' {result = val}
  | COMPONENT identifier local_generic_clause END COMPONENT
     component_simple_name ';' {result = val}
  | COMPONENT identifier IS END COMPONENT component_simple_name ';' {result =
     val}
  | COMPONENT identifier END COMPONENT component_simple_name ';' {result = val
     }
  | COMPONENT identifier IS local_generic_clause formal_port_clause END
     COMPONENT ';' {result = val}
  | COMPONENT identifier local_generic_clause formal_port_clause END COMPONENT
      ';' {result = val}
  | COMPONENT identifier IS formal_port_clause END COMPONENT ';' {result = val
     }
  | COMPONENT identifier formal_port_clause END COMPONENT ';' {result = val}
  | COMPONENT identifier IS local_generic_clause END COMPONENT ';' {result =
     val}
  | COMPONENT identifier local_generic_clause END COMPONENT ';' {result = val}
```

```
  | COMPONENT identifier IS END COMPONENT ';' {result = val}
  | COMPONENT identifier END COMPONENT ';' {result = val}

component_instantiation_statement :
  attribute_simple_name ':' instantiated_unit generic_map_aspect
     port_map_aspect ';' {result = ComponentInstantiation.new(val[0], val[2],
     val[3], val[4]);}
  | attribute_simple_name ':' instantiated_unit port_map_aspect ';' {result =
     ComponentInstantiation.new(val[0], val[2], nil, val[3]);}
  | attribute_simple_name ':' instantiated_unit generic_map_aspect ';' {result
      = ComponentInstantiation.new(val[0], val[2], val[3], nil); }
  | attribute_simple_name ':' instantiated_unit ';' {result =
     ComponentInstantiation.new(val[0], val[2], nil, nil); }

component_specification :
  instantiation_list ':' type_name {result = val;}

composite_type_definition :
  array_type_definition {result = val}
  | record_type_definition {result = val}

concurrent_assertion_statement :
  LABEL ':' POSTPONED assertion ';' {result = val}
  | POSTPONED assertion ';' {result = val}
  | LABEL ':' assertion ';' {result = val}
  | assertion ';' {result = val}

concurrent_procedure_call_statement :
  LABEL ':' POSTPONED procedure_call ';' {result = val}
  | POSTPONED procedure_call ';' {result = val}
  | LABEL ':' procedure_call ';' {result = val}
  | procedure_call ';' {result = val}

concurrent_signal_assignment_statement :
  LABEL ':' POSTPONED conditional_signal_assignment {result = val}
  | POSTPONED conditional_signal_assignment {result = val}
  | LABEL ':' conditional_signal_assignment {result = val}
  | conditional_signal_assignment {result = val}
  | LABEL ':' POSTPONED selected_signal_assignment {result = val}
  | POSTPONED selected_signal_assignment {result = val}
  | LABEL ':' selected_signal_assignment {result = val}
  | selected_signal_assignment {result = val}

concurrent_statement :
  block_statement {result = val; }
  | component_instantiation_statement {result = val;}
  | concurrent_procedure_call_statement {result = val; }
  | concurrent_assertion_statement {result = val; }
  | concurrent_signal_assignment_statement {result = val;}
  | process_statement {result = val[0]}
  | generate_statement {result = val;}
  | signal_assignment_statement {result = val[0]}
  |conditional_signal_assignment {result = val[0]}
```

```
condition :
  boolean_expression {result = val;}
  | function_call {result = ConditionalStatement.new(val[0], nil, nil);}
  | expression relational_operator expression {result=ConditionalStatement.new
     (val[0], val[1], val[2]);val;}
  | expression {result=ConditionalStatement.new(val[0], nil, nil);}
  | relation { val = val.flatten; result = ConditionalStatement.new(val[0],
     val[1], val[2]);val; }

condition_clause :
  UNTIL condition {result = val}

conditional_signal_assignment :
  target '<=' options conditional_waveforms ';' {result = val.flatten.join(' '
     )}
  | target '<=' conditional_waveforms ';'{result = val.flatten.join(' ')}
  | function_call '<=' conditional_waveforms ';' {result = val.flatten.join('
     ')}

conditional_waveforms :
  conditional_waveforms_loop0 waveform WHEN condition {result = val; ; }
  | conditional_waveforms_loop0 waveform {result = val; }
  | conditional_waveforms_loop0 {result = val; }

conditional_waveforms_loop0 :
  waveform WHEN condition ELSE conditional_waveforms_loop0 {result = val.
     flatten}
  | waveform {result = val[0]}
  | {result = val;}

configuration_declaration :
  CONFIGURATION identifier OF type_name IS configuration_declarative_part
     block_configuration END CONFIGURATION configuration_simple_name ';' {
     result = val}
  | CONFIGURATION identifier OF type_name IS configuration_declarative_part
     block_configuration END configuration_simple_name ';' {result = val}
  | CONFIGURATION identifier OF type_name IS configuration_declarative_part
     block_configuration END CONFIGURATION ';' {result = val}
  | CONFIGURATION identifier OF type_name IS configuration_declarative_part
     block_configuration END ';' {result = val}

configuration_declarative_item :
  use_clause {result = val}
  | attribute_specification {result = val}
  | group_declaration {result = val}

configuration_declarative_part :
  configuration_declarative_part_loop0 {result = val}

configuration_declarative_part_loop0 :
  configuration_declarative_item configuration_declarative_part_loop0 {result
```

```
      = val}
  | {result = val}

configuration_item :
  block_configuration {result = val}
  | component_configuration {result = val}

configuration_specification :
  FOR component_specification binding_indication ';' {result = val}

constant_declaration :
  CONSTANT identifier_list ':' subtype_indication ':=' expression ';' {result
      = ConstantDeclaration.new(val[1], val[3], val[5]); }
  | CONSTANT identifier_list ':' subtype_indication ';' {result =
      ConstantDeclaration.new(val[1], val[3], nil);}

constrained_array_definition :
  ARRAY index_constraint OF element_subtype_indication {result = val}

constraint :
  range_constraint {result = val}
  | index_constraint {result = val}

context_clause :
  context_clause_loop0 {result = val}

context_clause_loop0 :
  context_item context_clause_loop0 {result = val}
  | {result = val}

context_item :
  library_clause {result = val.flatten;}
  | use_clause {result = val.flatten;}

declaration :
  type_declaration {result = val}
  | subtype_declaration {result = val}
  | object_declaration {result = val}
  | interface_declaration {result = val}
  | alias_declaration {result = val}
  | attribute_declaration {result = val}
  | component_declaration {result = val}
  | group_template_declaration {result = val}
  | group_declaration {result = val}
  | entity_declaration {result = val}
  | configuration_declaration {result = val}
  | subprogram_declaration {result = val}
  | package_declaration {result = val}

delay_mechanism :
  TRANSPORT {result = val}
  | REJECT time_expression INERTIAL {result = val}
  | INERTIAL {result = val}
```

```
design_file :
  design_unit design_file_loop0 {result = DesignFile.new(val.flatten)}

design_file_loop0 :
  design_unit design_file_loop0 {result = val}
  | {result = val}

design_unit :
  context_clause library_unit {result = val}

designator :
  identifier {result = val}
  | operator_symbol {result = val}

direction :
  TO {result = val;}
  | DOWNTO {result = val}

disconnection_specification :
  DISCONNECT guarded_signal_specification AFTER time_expression ';' {result =
      val}

discrete_range :
  discrete_subtype_indication {result = val}
  | range_expression {result = val}

element_association :
  choices '=>' expression {val[2] = val[2].as_value;result =
      ElementAssociation.new(val[0], val[2]);}
  | expression {result = ElementAssociation.new([val[0]], val[2]);}

element_declaration :
  identifier_list ':' element_subtype_definition ';' {result = val}

element_subtype_definition :
  subtype_indication {result = val}

entity_aspect :
  ENTITY type_name '(' architecture_identifier ')' {result = val}
  | ENTITY type_name {result = val}
  | CONFIGURATION configuration_name {result = val}
  | OPEN {result = val}

entity_class :
  ENTITY {result = val}
  | ARCHITECTURE {result = val}
  | CONFIGURATION {result = val}
  | PROCEDURE {result = val}
  | FUNCTION {result = val}
  | PACKAGE {result = val}
  | TYPE {result = val}
  | SUBTYPE {result = val}
```

```
  | CONSTANT {result = val}
  | SIGNAL {result = val}
  | VARIABLE {result = val}
  | COMPONENT {result = val}
  | LABEL {result = val}
  | LITERAL {result = val}
  | UNITS {result = val}
  | GROUP {result = val}
  | FILE {result = val}

entity_class_entry :
  entity_class '<>' {result = val}
  | entity_class {result = val}

entity_class_entry_list :
  entity_class_entry entity_class_entry_list_loop0 {result = val}

entity_class_entry_list_loop0 :
  ',' entity_class_entry entity_class_entry_list_loop0 {result = val}
  | {result = val}

entity_declaration :
  ENTITY identifier IS entity_header entity_declarative_part BEGIN
      entity_statement_part END ENTITY type_name ';' { InitializeRepository.add
      (val[1]) ;result = EntityDeclaration.new(val[1], val[3], val[4].flatten)
      ;}
  | ENTITY identifier IS entity_header entity_declarative_part END ENTITY
      type_name ';' { InitializeRepository.add(val[1]) ;result =
      EntityDeclaration.new(val[1], val[3], val[4].flatten);}
  | ENTITY identifier IS entity_header entity_declarative_part BEGIN
      entity_statement_part END type_name ';' { InitializeRepository.add(val
      [1]) ;result = EntityDeclaration.new(val[1], val[3], val[4].flatten);}
  | ENTITY identifier IS entity_header entity_declarative_part END type_name '
      ;' { InitializeRepository.add(val[1]) ;result = EntityDeclaration.new(val
      [1], val[3], val[4].flatten);}
  | ENTITY identifier IS entity_header entity_declarative_part BEGIN
      entity_statement_part END ENTITY ';' { InitializeRepository.add(val[1]) ;
      result = EntityDeclaration.new(val[1], val[3], val[4].flatten);}
  | ENTITY identifier IS entity_header entity_declarative_part END ENTITY ';'
      { InitializeRepository.add(val[1]) ;result = EntityDeclaration.new(val
      [1], val[3], val[4].flatten);}
  | ENTITY identifier IS entity_header entity_declarative_part BEGIN
      entity_statement_part END ';' { InitializeRepository.add(val[1]) ;result
      = EntityDeclaration.new(val[1], val[3], val[4].flatten);}
  | ENTITY identifier IS entity_header entity_declarative_part END ';' {
      InitializeRepository.add(val[1]) ;result = EntityDeclaration.new(val[1],
      val[3], val[4].flatten);}

entity_declarative_item :
  subprogram_declaration {result = val}
  | subprogram_body {result = val}
  | type_declaration {result = val}
  | subtype_declaration {result = val}
```

```
  | constant_declaration {result = val[0] }
  | signal_declaration {result = va[0]}
  | shared_variable_declaration {result = val}
  | file_declaration {result = val}
  | alias_declaration {result = val}
  | attribute_declaration {result = val}
  | attribute_specification {result = val}
  | disconnection_specification {result = val}
  | use_clause {result = val}
  | group_template_declaration {result = val}
  | group_declaration {result = val}

entity_declarative_part :
  entity_declarative_part_loop0 {result = val}

entity_declarative_part_loop0 :
  entity_declarative_item entity_declarative_part_loop0 {result = val}
  | {result = val}

entity_designator :
  entity_tag signature {result = val}
  | entity_tag {result = val}

entity_header :
  formal_generic_clause formal_port_clause {result = val}
  | formal_port_clause {result = val; }
  | formal_generic_clause {result = val;}
  | {result = val}

formal_generic_clause :
  generic_clause {result = val[0];}
entity_name_list :
  entity_designator entity_name_list_loop0 {result = val;}
  | OTHERS {result = val;}
  | ALL {result = val}

entity_name_list_loop0 :
  ',' entity_designator entity_name_list_loop0 {result = val}
  | {result = val}

entity_specification :
  entity_name_list ':' entity_class {result = val;}

entity_statement :
  concurrent_assertion_statement {result = val}
  | passive_concurrent_procedure_call_statement {result = val}
  | passive_process_statement {result = val}

entity_statement_part :
  entity_statement_part_loop0 {result = val}

entity_statement_part_loop0 :
  entity_statement entity_statement_part_loop0 {result = val}
```

```
  | {result = val}

entity_tag :
  label {result = val}
  | CHARACTER_LITERAL {result = CharacterLiteral.new(val[0]);
     ConstantValueRepository.add(result);}
  | operator_symbol {result = val}

enumeration_literal :
  identifier {result = val[0];}
  | CHARACTER_LITERAL {result = CharacterLiteral.new(val[0]);
     ConstantValueRepository.add(result); }

enumeration_type_definition :
  '(' enumeration_literal enumeration_type_definition_loop0 ')' {result = val}

enumeration_type_definition_loop0 :
  ',' enumeration_literal enumeration_type_definition_loop0 {result = val}
  | {result = val}

exit_statement :
  LABEL ':' EXIT loop_label WHEN condition ';' {result = val}
  | EXIT loop_label WHEN condition ';' {result = val}
  | LABEL ':' EXIT WHEN condition ';' {result = val}
  | EXIT WHEN condition ';' {result = val}
  | LABEL ':' EXIT loop_label ';' {result = val}
  | EXIT loop_label ';' {result = val}
  | LABEL ':' EXIT ';' {result = val}
  | EXIT ';' {result = val}

expression :
  relation expression_loop0 {result = Expression.new(val.flatten);}
  | STRING_LITERAL {result = StringLiteral.new(val[0]);}
  | relation expression_loop1 {result = Expression.new(val.flatten);}
  | relation expression_loop2 {result = Expression.new(val.flatten);}
  | relation NAND relation {result = Expression.new(val.flatten);}
  | relation NOR relation {result = Expression.new(val.flatten);}
  | relation expression_loop3 {result = Expression.new(val.flatten);}

expression_loop0 :
  AND relation expression_loop0 {result = val;}
  | {result = val}

expression_loop1 :
  OR relation expression_loop1 {result = val}
  | {result = val}

expression_loop2 :
  XOR relation expression_loop2 {result = val}
  | {result = val}

expression_loop3 :
  XNOR relation expression_loop3 {result = val}
```

```
  | {result = val}

extended_digit :
  digit {result = val}
  | letter {result = val}

factor :
  primary '**' primary {result = val;}
  | primary {result = val[0];}
  | ABS primary {result = val}
  | NOT primary {result = val; }


file_declaration :
  FILE identifier_list ':' subtype_indication file_open_information ';' {
     result = val}
  | FILE identifier_list ':' subtype_indication ';' {result = val}

file_logical_name :
  string_expression {result = val}

file_open_information :
  OPEN file_open_kind_expression IS file_logical_name {result = val}
  | IS file_logical_name {result = val}

file_type_definition :
  FILE OF type_mark floating_type_definition ':=' range_constraint {result =
     val}

formal_designator :
  type_name {result = val[0]}
  | identifier {result = val[0]}

formal_parameter_list :
  parameter_interface_list {result = val}

function_name :
  identifier {result = val}
formal_part :
  formal_designator {result = val[0]}
  | function_call {result = val[0];}
  | type_mark '(' formal_designator ')' {result = val}


formal_port_clause :
  port_clause {result = val[0];}

full_type_declaration :
  TYPE identifier IS type_definition ';' {result = val}

function_call :
  identifier '(' identifier_list ')' {result = FunctionCall.new(val[0], val
     [2]);}
```

```
  | identifier '(' numeric_literal ')' {result = FunctionCall.new(val[0], val
      [2]);}
  | identifier '(' range_expression ')'{val[2].ignore_braces = true;result =
      FunctionCall.new(val[0], val[2]);}
  | identifier '(' simple_expression ')'{result = FunctionCall.new(val[0], val
      [2]);}


generate_statement :
  generate_label ':' generation_scheme GENERATE generate_statement_loop0 BEGIN
      generate_statement_loop1 END GENERATE generate_label ';' {result = val}
  | generate_label ':' generation_scheme GENERATE generate_statement_loop1 END
      GENERATE generate_label ';' {result = val}
  | generate_label ':' generation_scheme GENERATE generate_statement_loop0
     BEGIN generate_statement_loop1 END GENERATE ';' {result = val}
  | generate_label ':' generation_scheme GENERATE generate_statement_loop1 END
      GENERATE ';' {result = val}

generate_statement_loop0 :
  block_declarative_item generate_statement_loop0 {result = val}
  | {result = val}

generate_statement_loop1 :
  concurrent_statement generate_statement_loop1 {result = val}
  | {result = val}

generation_scheme :
  FOR generate_parameter_specification {result = val}
  | IF condition {result = val;}

generic_association_list :
  association_list {result = val[0];}

generic_clause :
  GENERIC '(' generic_list ')' ';' {result = GenericClause.new(val[2]);}

generic_list :
  port_interface_list {result = val[0]; }

generic_map_aspect :
  GENERIC MAP '(' generic_association_list ')' {val[3].elements.each {|el| el.
     actual_part = el.actual_part.as_value }; result = PortMap.new('GENERIC',
     val[3]);}

group_constituent :
  name {result = val}
  | CHARACTER_LITERAL {result = CharacterLiteral.new(val[0]);
     ConstantValueRepository.add(result);}

group_constituent_list :
  group_constituent group_constituent_list_loop0 {result = val}

group_constituent_list_loop0 :
```

```
  ',' group_constituent group_constituent_list_loop0 {result = val}
  | {result = val}

group_declaration :
  GROUP identifier ':' group_template_name '(' group_constituent_list ')' ';'
     {result = val}

group_template_declaration :
  GROUP identifier IS '(' entity_class_entry_list ')' ';' {result = val}

guarded_signal_specification :
  guarded_signal_list ':' type_mark {result = val}

identifier :
  BASIC_IDENTIFIER {result = Identifier.new(val[0]);}
  | EXTENDED_IDENTIFIER {result = Identifier.new(val[0])}

identifier_list :
  identifier identifier_list_loop0 {result = val = IdentifierList.new(val.
     flatten); InitializeRepository.add(result.identifiers) }

identifier_list_loop0 :
  ',' identifier identifier_list_loop0 {result = val - [',']}
  | {result = val}

if_statement :
  if_label ':' IF condition THEN sequence_of_statements if_statement_loop0
     ELSE sequence_of_statements END IF if_label ';' {result = IfStatement.new
     (val[0], val[3], val[5..6], val[8], val[9...val.length-1])}
  | IF condition THEN sequence_of_statements if_statement_loop0 ELSE
     sequence_of_statements END IF if_label ';' {result = IfStatement.new(nil,
      val[1], val[3..4], val[6], val[7...val.length-1])}
  | if_label ':' IF condition THEN sequence_of_statements if_statement_loop0
     END IF if_label ';' {result = IfStatement.new(val[0], val[3], val[5..6],
     nil, val[9...val.length-1])}
  | IF condition THEN sequence_of_statements if_statement_loop0 END IF
     if_label ';' {result = IfStatement.new(nil, val[1], val[3..4], nil, val
     [5...val.length-1])}
  | if_label ':' IF condition THEN sequence_of_statements if_statement_loop0
     ELSE sequence_of_statements END IF ';' {result = IfStatement.new(val[0],
     val[3], val[5..6], val[8], val[9...val.length-1])}
  | IF condition THEN sequence_of_statements if_statement_loop0 ELSE
     sequence_of_statements END IF ';' {result = IfStatement.new(nil, val[1],
     val[3..4], val[6], val[7...val.length-1])}
  | if_label ':' IF condition THEN sequence_of_statements if_statement_loop0
     END IF ';' {result = IfStatement.new(val[0], val[3], val[5..6], nil, val
     [7...val.length-1])}
  | IF condition THEN sequence_of_statements if_statement_loop0 END IF ';' {
     result = IfStatement.new(nil, val[1], val[3..4], nil, val[5...val.length
     -1])}
  | IF '(' condition ')' THEN sequence_of_statements if_statement_loop0 END IF
      ';' {result = IfStatement.new(nil, val[2], val[5..6], nil, val[7...val.
     length-1]);}
```

```
  | IF '(' condition ')' THEN sequence_of_statements if_statement_loop0 ELSE
      sequence_of_statements END IF ';' {result = IfStatement.new(nil, val[2],
      val[5..6], val[8], val[9...val.length-1]);}

if_statement_loop0 :
  ELSIF condition THEN sequence_of_statements if_statement_loop0 {result = val
      ;}
  | {result = val}

incomplete_type_declaration :
  TYPE identifier ';' {result = val}

index_constraint :
  '(' discrete_range index_constraint_loop0 ')' {result = val}

index_constraint_loop0 :
  ',' discrete_range index_constraint_loop0 {result = val}
  | {result = val}

index_specification :
  discrete_range {result = val}
  | static_expression {result = val}

index_subtype_definition :
  type_mark RANGE '<>' {result = val}

indexed_name :
  prefix '(' expression indexed_name_loop0 ')' {result = val}

indexed_name_loop0 :
  ',' expression indexed_name_loop0 {result = val}
  | {result = val}

instantiated_unit :
  COMPONENT type_name {result = val}
  | type_name {result = val[0];}
  | identifier {result = val[0]; }
  | ENTITY type_name '(' architecture_identifier ')' {result = val}
  | ENTITY type_name {result = val}
  | CONFIGURATION configuration_name {result = val}

instantiation_list :
  attribute_simple_name instantiation_list_loop0 {result = val;}
  | OTHERS {result = val}
  | ALL {result = val}

instantiation_list_loop0 :
  ',' attribute_simple_name instantiation_list_loop0 {result = val}
  | {result = val}

integer_type_definition :
  range_constraint {result = val}
```

```
interface_constant_declaration :
  CONSTANT identifier_list ':' IN subtype_indication ':=' static_expression {
     result = SignalDeclaration.new(val[1], val[3][0]); }
  | CONSTANT identifier_list ':' subtype_indication ':=' static_expression {
     result = SignalDeclaration.new(val[1], val[3][0]); }
  | CONSTANT identifier_list ':' IN subtype_indication {result =
     SignalDeclaration.new(val[1], val[3][0]); }
  | CONSTANT identifier_list ':' subtype_indication {result =
     SignalDeclaration.new(val[1], nil); }


interface_declaration :
  interface_signal_declaration {result = val}
  | interface_variable_declaration {result = val}
  | interface_file_declaration {result = val}
  | interface_constant_declaration {result = val;}

interface_element :
  interface_declaration {result = val[0];}

interface_file_declaration :
  FILE identifier_list ':' subtype_indication {result = val}

interface_list :
  interface_element interface_list_loop0 {result = val.flatten; result.last.
     type.append_semicolon = false; }

interface_list_loop0 :
  ';' interface_element interface_list_loop0 {result = val[1..2]}
  | {result = val}

interface_signal_declaration :
  SIGNAL identifier_list ':' mode subtype_indication BUS ':='
     static_expression {val[4].append_semicolon = true ;result =
     SignalDeclaration.new(val[0],val[1], val[3], val[4], val[7]); }
  | identifier_list ':' mode subtype_indication BUS ':=' static_expression {
     val[3].append_semicolon = true ;result = SignalDeclaration.new(nil, val
     [0], val[2], val[3], val[6]); }
  | SIGNAL identifier_list ':' subtype_indication BUS ':=' static_expression {
     val[3].append_semicolon = true ;result = SignalDeclaration.new(val[0],val
     [1], nil, val[3], val[6]); }
  | identifier_list ':' subtype_indication BUS ':=' static_expression {val[2].
     append_semicolon = true ;result = SignalDeclaration.new(nil, val[0], nil,
      val[2], val[5] ); }
  | SIGNAL identifier_list ':' mode subtype_indication ':=' static_expression
     {val[4].append_semicolon = true ;result = SignalDeclaration.new(val[0],
     val[1], val[3], val[4], val[6]); }
  | identifier_list ':' mode subtype_indication ':=' static_expression {val
     [3].append_semicolon = true ;result = SignalDeclaration.new(nil, val[0],
     val[2], val[3], val[5]); }
  | SIGNAL identifier_list ':' subtype_indication ':=' static_expression {val
     [3].append_semicolon = true ;result = SignalDeclaration.new(val[0], val
     [1], nil, val[3],val[5]); }
```

```
  | identifier_list ':' subtype_indication ':=' static_expression {val[2].
     append_semicolon = true ;result = SignalDeclaration.new(nil, val[0], nil,
      val[2], val[4]); }
  | SIGNAL identifier_list ':' mode subtype_indication BUS {val[4].
     append_semicolon = true ;result = SignalDeclaration.new(val[0], val[1],
     val[3], val[4]); }
  | identifier_list ':' mode subtype_indication BUS {val[3].append_semicolon =
      true ;result = SignalDeclaration.new(nil, val[0], val[2], val[3]); }
  | SIGNAL identifier_list ':' subtype_indication BUS {val[3].append_semicolon
      = true ;result = SignalDeclaration.new(val[0], val[1], nil, val[3]); }
  | identifier_list ':' subtype_indication BUS {val[2].append_semicolon = true
      ;result = SignalDeclaration.new(nil, val[0], nil, val[2]); }
  | SIGNAL identifier_list ':' mode subtype_indication {val[4].
     append_semicolon = true ;result = SignalDeclaration.new(val[0], val[1],
     val[3], val[4]); }
  | identifier_list ':' mode subtype_indication {val[3].append_semicolon =
     true ;result = SignalDeclaration.new(nil, val[0], val[2], val[3]);}
  | SIGNAL identifier_list ':' subtype_indication {val[3].append_semicolon =
     true ;result = SignalDeclaration.new(val[0], val[1], nil, val[3]); }
  | identifier_list ':' subtype_indication {val[2].append_semicolon = true ;
     result = SignalDeclaration.new(nil, val[0], nil, val[2]); }

interface_variable_declaration :
  VARIABLE identifier_list ':' mode subtype_indication ':=' static_expression
     {result = val}
  | identifier_list ':' mode subtype_indication ':=' static_expression {result
      = val; }
  | VARIABLE identifier_list ':' subtype_indication ':=' static_expression {
     result = val; }
  | identifier_list ':' subtype_indication ':=' static_expression {result =
     val; }
  | VARIABLE identifier_list ':' mode subtype_indication {result = val; }
  | identifier_list ':' mode subtype_indication {result = val; }
  | VARIABLE identifier_list ':' subtype_indication {result = val}
  | identifier_list ':' subtype_indication {result = val; }

iteration_scheme :
  WHILE condition {result = val}
  | FOR loop_parameter_specification {result = val}

label :
  identifier {result = val[0]; }

library_clause :
  LIBRARY logical_name_list ';' {result = LibraryWrapper.new(val[1].flatten);}

library_unit :
  primary_unit {result = val}
  | secondary_unit {result = val}

literal_expression :
  numeric_literal {result = val[0]}
  | enumeration_literal {result = val[0]}
```

```
  | STRING_LITERAL {result = StringLiteral.new(val[0]);}
  | BIT_STRING_LITERAL {result = val[0];}
  | NULL {result = val[0];}

local_generic_clause :
  generic_clause {result = val}

logical_name_list :
  label logical_name_list_loop0 {result = val}

logical_name_list_loop0 :
  ',' label logical_name_list_loop0 {result = val}
  | {result = val}

logical_operator :
  AND {result = val}
  | OR {result = val}
  | NAND {result = val}
  | NOR {result = val}
  | XOR {result = val}
  | XNOR {result = val}

loop_statement :
  loop_label ':' iteration_scheme LOOP sequence_of_statements END LOOP
     loop_label ';' {result = val;}
  | iteration_scheme LOOP sequence_of_statements END LOOP loop_label ';' {
     result = val;}
  | loop_label ':' LOOP sequence_of_statements END LOOP loop_label ';' {result
      = val;}
  | LOOP sequence_of_statements END LOOP loop_label ';' {result = val;}
  | loop_label ':' iteration_scheme LOOP sequence_of_statements END LOOP ';' {
     result = val;}
  | iteration_scheme LOOP sequence_of_statements END LOOP ';' {result = val;}
  | loop_label ':' LOOP sequence_of_statements END LOOP ';' {result = val;}
  | LOOP sequence_of_statements END LOOP ';' {result = val;}

miscellaneous_operator :
  '**' {result = val}
  | ABS {result = val}
  | NOT {result = val}

mode :
  IN {result = val[0]}
  | OUT {result = val[0]}
  | INOUT {result = val[0]}
  | BUFFER {result = val[0]}
  | LINKAGE {result = val[0]}

multiplying_operator :
  '*' {result = val}
  | '/' {result = val}
  | MOD {result = val}
  | REM {result = val}
```

```
name :
  label {result = val[0]}
  | operator_symbol {result = val[0]}
  | selected_name {result = val[0]}
  | indexed_name {result = val[0]}
  | slice_name {result = val[0]}
  | attribute_name {result = val[0]}

next_statement :
  LABEL ':' NEXT loop_label WHEN condition ';' {result = val}
  | NEXT loop_label WHEN condition ';' {result = val}
  | LABEL ':' NEXT WHEN condition ';' {result = val}
  | NEXT WHEN condition ';' {result = val}
  | LABEL ':' NEXT loop_label ';' {result = val}
  | NEXT loop_label ';' {result = val}
  | LABEL ':' NEXT ';' {result = val}
  | NEXT ';' {result = val}

null_statement :
  LABEL ':' NULL ';' {result = val}
  | NULL ';' {result = val}

numeric_literal :
  abstract_literal {result = val[0]}
  | physical_literal {result = val}

object_declaration :
  constant_declaration {result = val[0]}
  | signal_declaration {result = val}
  | variable_declaration {result = val}
  | file_declaration {result = val}

operator_symbol :
  STRING_LITERAL {result = StringLiteral.new(val[0]);}

options :
  GUARDED delay_mechanism {result = val}
  | delay_mechanism {result = val}
  | GUARDED {result = val}
  | {result = val}

package_body :
  PACKAGE BODY package_simple_name IS package_body_declarative_part END
     PACKAGE BODY package_simple_name ';' {result = val}
  | PACKAGE BODY package_simple_name IS package_body_declarative_part END
     package_simple_name ';' {result = val}
  | PACKAGE BODY package_simple_name IS package_body_declarative_part END
     PACKAGE BODY ';' {result = val}
  | PACKAGE BODY package_simple_name IS package_body_declarative_part END ';'
     {result = val}

package_body_declarative_item :
```

```
  subprogram_declaration {result = val}
  | subprogram_body {result = val}
  | type_declaration {result = val}
  | subtype_declaration {result = val}
  | constant_declaration {result = val[0]}
  | shared_variable_declaration {result = val}
  | file_declaration {result = val}
  | alias_declaration {result = val}
  | use_clause {result = val}
  | group_template_declaration {result = val}
  | group_declaration {result = val}

package_body_declarative_part :
  package_body_declarative_part_loop0 {result = val}

package_body_declarative_part_loop0 :
  package_body_declarative_item package_body_declarative_part_loop0 {result =
     val}
  | {result = val}

package_declaration :
  PACKAGE identifier IS package_declarative_part END PACKAGE
     package_simple_name ';' {result = val}
  | PACKAGE identifier IS package_declarative_part END package_simple_name ';'
      {result = val}
  | PACKAGE identifier IS package_declarative_part END PACKAGE ';' {result =
     val}
  | PACKAGE identifier IS package_declarative_part END ';' {result = val}

package_declarative_item :
  subprogram_declaration {result = val}
  | type_declaration {result = val}
  | subtype_declaration {result = val}
  | constant_declaration {result = val[0]}
  | signal_declaration {result = val}
  | shared_variable_declaration {result = val}
  | file_declaration {result = val}
  | alias_declaration {result = val}
  | component_declaration {result = val}
  | attribute_declaration {result = val}
  | attribute_specification {result = val}
  | disconnection_specification {result = val}
  | use_clause {result = val}
  | group_template_declaration {result = val}
  | group_declaration {result = val}

package_declarative_part :
  package_declarative_part_loop0 {result = val}

package_declarative_part_loop0 :
  package_declarative_item package_declarative_part_loop0 {result = val}
  | {result = val}
```

```
parameter_specification :
  identifier IN discrete_range {result = val}

physical_literal :
  abstract_literal unit_name {result = val}
  | unit_name {result = val}

physical_type_definition :
  range_constraint UNITS base_unit_declaration physical_type_definition_loop0
      END UNITS physical_type_simple_name {result = val}
  | range_constraint UNITS base_unit_declaration
      physical_type_definition_loop0 END UNITS {result = val}

physical_type_definition_loop0 :
  secondary_unit_declaration physical_type_definition_loop0 {result = val}
  | {result = val}

port_clause :
  PORT '(' generic_list ')' ';' {result = PortClause.new(val[2].flatten);}

port_interface_list :
  interface_list {result = val[0]}

port_map_aspect :
  PORT MAP '(' generic_association_list ')' {result = PortMap.new('PORT', val
      [3]);}

prefix :
  name {result = val[0]}
  | function_call {result = val[0];}

primary :
  name {result = val[0];}
  | literal_expression {result = val[0];}
  | aggregate {result = val[0];}
  | function_call {result = val[0];}
  | qualified_expression {result = val;}
  | type_conversion {result = val;}
  | allocator {result = val;}
  | '(' expression ')' {result = val;}

primary_unit :
  entity_declaration {result = val}
  | configuration_declaration {result = val}
  | package_declaration {result = val}

procedure_call :
  procedure_name '(' actual_parameter_part ')' {result = val}
  | procedure_name {result = val}

procedure_call_statement :
  LABEL ':' procedure_call ';' {result = val}
  | procedure_call ';' {result = val}
```

```
process_declarative_item :
  subprogram_declaration {result = val}
  | subprogram_body {result = val}
  | type_declaration {result = val}
  | subtype_declaration {result = val}
  | constant_declaration {result = val[0]}
  | variable_declaration {result = val}
  | file_declaration {result = val}
  | alias_declaration {result = val}
  | attribute_declaration {result = val}
  | attribute_specification {result = val}
  | use_clause {result = val}
  | group_template_declaration {result = val}
  | group_declaration {result = val}

process_declarative_part :
  process_declarative_part_loop0 {result = val}

process_declarative_part_loop0 :
  process_declarative_item process_declarative_part_loop0 {result = val}
  | {result = val}


process_statement :
  PROCESS_NAME ':' POSTPONED PROCESS '(' sensitivity_list ')' IS
     process_declarative_part BEGIN process_statement_part END POSTPONED
     PROCESS identifier ';' {result = ProcessDeclaration.new(nil, val[0], ); }
  | POSTPONED PROCESS '(' sensitivity_list ')' IS process_declarative_part
     BEGIN process_statement_part END POSTPONED PROCESS identifier ';' {result
      = ProcessDeclaration.new(nil, val[0], ); }
  | PROCESS_NAME ':' PROCESS '(' sensitivity_list ')' IS
     process_declarative_part BEGIN process_statement_part END POSTPONED
     PROCESS identifier ';' {result = ProcessDeclaration.new(nil, val[0], ); }
  | PROCESS '(' sensitivity_list ')' IS process_declarative_part BEGIN
     process_statement_part END POSTPONED PROCESS identifier ';' {result =
     ProcessDeclaration.new(nil, val[0], ); }
  | PROCESS_NAME ':' POSTPONED PROCESS IS process_declarative_part BEGIN
     process_statement_part END POSTPONED PROCESS identifier ';' {result =
     ProcessDeclaration.new(nil, val[0], ); }
  | POSTPONED PROCESS IS process_declarative_part BEGIN process_statement_part
      END POSTPONED PROCESS identifier ';' {result = ProcessDeclaration.new(
     nil, val[0], ); }
  | PROCESS_NAME ':' PROCESS IS process_declarative_part BEGIN
     process_statement_part END POSTPONED PROCESS identifier ';' {result =
     ProcessDeclaration.new(nil, val[0], ); }
  | PROCESS IS process_declarative_part BEGIN process_statement_part END
     POSTPONED PROCESS identifier ';' {result = ProcessDeclaration.new(nil,
     val[0], ); }
  | PROCESS_NAME ':' POSTPONED PROCESS '(' sensitivity_list ')'
     process_declarative_part BEGIN process_statement_part END POSTPONED
     PROCESS identifier ';' {result = ProcessDeclaration.new(nil, val[0], ); }
  | POSTPONED PROCESS '(' sensitivity_list ')' process_declarative_part BEGIN
```

```
   process_statement_part END POSTPONED PROCESS identifier ';' {result =
   ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' PROCESS '(' sensitivity_list ')' process_declarative_part
    BEGIN process_statement_part END POSTPONED PROCESS identifier ';' {
   result = ProcessDeclaration.new(nil, val[0], ); }
| PROCESS '(' sensitivity_list ')' process_declarative_part BEGIN
   process_statement_part END POSTPONED PROCESS identifier ';' {result =
   ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' POSTPONED PROCESS process_declarative_part BEGIN
   process_statement_part END POSTPONED PROCESS identifier ';' {result =
   ProcessDeclaration.new(nil, val[0], ); }
| POSTPONED PROCESS process_declarative_part BEGIN process_statement_part
   END POSTPONED PROCESS identifier ';' {result = ProcessDeclaration.new(nil
   , val[0], ); }
| PROCESS_NAME ':' PROCESS process_declarative_part BEGIN
   process_statement_part END POSTPONED PROCESS identifier ';' {result =
   ProcessDeclaration.new(nil, val[0], ); }
| PROCESS process_declarative_part BEGIN process_statement_part END
   POSTPONED PROCESS identifier ';' {result = ProcessDeclaration.new(nil,
   val[0], ); }
| PROCESS_NAME ':' POSTPONED PROCESS '(' sensitivity_list ')' IS
   process_declarative_part BEGIN process_statement_part END PROCESS
   identifier ';' {result = ProcessDeclaration.new(nil, val[0], ); }
| POSTPONED PROCESS '(' sensitivity_list ')' IS process_declarative_part
   BEGIN process_statement_part END PROCESS identifier ';' {result =
   ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' PROCESS '(' sensitivity_list ')' IS
   process_declarative_part BEGIN process_statement_part END PROCESS
   identifier ';' {result = ProcessDeclaration.new(nil, val[0], ); }
| PROCESS '(' sensitivity_list ')' IS process_declarative_part BEGIN
   process_statement_part END PROCESS identifier ';' {result =
   ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' POSTPONED PROCESS IS process_declarative_part BEGIN
   process_statement_part END PROCESS identifier ';' {result =
   ProcessDeclaration.new(nil, val[0], ); }
| POSTPONED PROCESS IS process_declarative_part BEGIN process_statement_part
    END PROCESS identifier ';' {result = ProcessDeclaration.new(nil, val[0],
    ); }
| PROCESS_NAME ':' PROCESS IS process_declarative_part BEGIN
   process_statement_part END PROCESS identifier ';' {result =
   ProcessDeclaration.new(nil, val[0], ); }
| PROCESS IS process_declarative_part BEGIN process_statement_part END
   PROCESS identifier ';' {result = ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' POSTPONED PROCESS '(' sensitivity_list ')'
   process_declarative_part BEGIN process_statement_part END PROCESS
   identifier ';' {result = ProcessDeclaration.new(nil, val[0], ); }
| POSTPONED PROCESS '(' sensitivity_list ')' process_declarative_part BEGIN
   process_statement_part END PROCESS identifier ';' {result =
   ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' PROCESS '(' sensitivity_list ')' process_declarative_part
    BEGIN process_statement_part END PROCESS identifier ';' {result =
   ProcessDeclaration.new(nil, val[0], ); }
| PROCESS '(' sensitivity_list ')' process_declarative_part BEGIN
```

```
    process_statement_part END PROCESS identifier ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' POSTPONED PROCESS process_declarative_part BEGIN
    process_statement_part END PROCESS identifier ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| POSTPONED PROCESS process_declarative_part BEGIN process_statement_part
    END PROCESS identifier ';' {result = ProcessDeclaration.new(nil, val[0],
    ); }
| PROCESS_NAME ':' PROCESS process_declarative_part BEGIN
    process_statement_part END PROCESS identifier ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| PROCESS process_declarative_part BEGIN process_statement_part END PROCESS
    identifier ';' {result = ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' POSTPONED PROCESS '(' sensitivity_list ')' IS
    process_declarative_part BEGIN process_statement_part END POSTPONED
    PROCESS ';' {result = ProcessDeclaration.new(nil, val[0], ); }
| POSTPONED PROCESS '(' sensitivity_list ')' IS process_declarative_part
    BEGIN process_statement_part END POSTPONED PROCESS ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' PROCESS '(' sensitivity_list ')' IS
    process_declarative_part BEGIN process_statement_part END POSTPONED
    PROCESS ';' {result = ProcessDeclaration.new(nil, val[0], ); }
| PROCESS '(' sensitivity_list ')' IS process_declarative_part BEGIN
    process_statement_part END POSTPONED PROCESS ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' POSTPONED PROCESS IS process_declarative_part BEGIN
    process_statement_part END POSTPONED PROCESS ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| POSTPONED PROCESS IS process_declarative_part BEGIN process_statement_part
     END POSTPONED PROCESS ';' {result = ProcessDeclaration.new(nil, val[0],
    ); }
| PROCESS_NAME ':' PROCESS IS process_declarative_part BEGIN
    process_statement_part END POSTPONED PROCESS ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| PROCESS IS process_declarative_part BEGIN process_statement_part END
    POSTPONED PROCESS ';' {result = ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' POSTPONED PROCESS '(' sensitivity_list ')'
    process_declarative_part BEGIN process_statement_part END POSTPONED
    PROCESS ';' {result = ProcessDeclaration.new(nil, val[0], ); }
| POSTPONED PROCESS '(' sensitivity_list ')' process_declarative_part BEGIN
    process_statement_part END POSTPONED PROCESS ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' PROCESS '(' sensitivity_list ')' process_declarative_part
     BEGIN process_statement_part END POSTPONED PROCESS ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| PROCESS '(' sensitivity_list ')' process_declarative_part BEGIN
    process_statement_part END POSTPONED PROCESS ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| PROCESS_NAME ':' POSTPONED PROCESS process_declarative_part BEGIN
    process_statement_part END POSTPONED PROCESS ';' {result =
    ProcessDeclaration.new(nil, val[0], ); }
| POSTPONED PROCESS process_declarative_part BEGIN process_statement_part
    END POSTPONED PROCESS ';' {result = ProcessDeclaration.new(nil, val[0], )
```

```
    ; }
  | PROCESS_NAME ':' PROCESS process_declarative_part BEGIN
     process_statement_part END POSTPONED PROCESS ';' {result =
     ProcessDeclaration.new(nil, val[0], ); }
  | PROCESS process_declarative_part BEGIN process_statement_part END
     POSTPONED PROCESS ';' {result = ProcessDeclaration.new(nil, val[0], ); }
  | PROCESS_NAME ':' POSTPONED PROCESS '(' sensitivity_list ')' IS
     process_declarative_part BEGIN process_statement_part END PROCESS ';' {
     result = ProcessDeclaration.new(nil, val[0], ); }
  | POSTPONED PROCESS '(' sensitivity_list ')' IS process_declarative_part
     BEGIN process_statement_part END PROCESS ';' {result = ProcessDeclaration
     .new(nil, val[0], ); }
  | PROCESS_NAME ':' PROCESS '(' sensitivity_list ')' IS
     process_declarative_part BEGIN process_statement_part END PROCESS ';' {
     result = ProcessDeclaration.new(nil, val[0], ); }
  | PROCESS '(' sensitivity_list ')' IS process_declarative_part BEGIN
     process_statement_part END PROCESS ';' {result = ProcessDeclaration.new(
     nil, val[0], ); }
  | PROCESS_NAME ':' POSTPONED PROCESS IS process_declarative_part BEGIN
     process_statement_part END PROCESS ';' {result = ProcessDeclaration.new(
     nil, val[0], ); }
  | POSTPONED PROCESS IS process_declarative_part BEGIN process_statement_part
      END PROCESS ';' {result = ProcessDeclaration.new(nil, val[0], ); }
  | PROCESS_NAME ':' PROCESS IS process_declarative_part BEGIN
     process_statement_part END PROCESS ';' {result = ProcessDeclaration.new(
     nil, val[0], ); }
  | PROCESS IS process_declarative_part BEGIN process_statement_part END
     PROCESS ';' {result = ProcessDeclaration.new(nil, val[0], ); }
  | PROCESS_NAME ':' POSTPONED PROCESS '(' sensitivity_list ')'
     process_declarative_part BEGIN process_statement_part END PROCESS ';' {
     result = ProcessDeclaration.new(nil, val[0], ); }
  | POSTPONED PROCESS '(' sensitivity_list ')' process_declarative_part BEGIN
     process_statement_part END PROCESS ';' {result = ProcessDeclaration.new(
     nil, val[0], ); }
  | PROCESS_NAME ':' PROCESS '(' sensitivity_list ')' process_declarative_part
      BEGIN process_statement_part END PROCESS ';' {result =
     ProcessDeclaration.new(nil, val[0], val[4], val[6], val[8], val[9...val.
     length-1]); }
  | PROCESS '(' sensitivity_list ')' BEGIN process_statement_part END PROCESS
     ';' {result = ProcessDeclaration.new(nil, nil, val[2], [], val[5], val
     [6...val.length-1]); }
  | PROCESS '(' sensitivity_list ')' process_declarative_part BEGIN
     process_statement_part END PROCESS ';' {result = ProcessDeclaration.new(
     nil, nil, val[2], val[4], val[6], val[7...val.length-1]); }
  | PROCESS_NAME ':' POSTPONED PROCESS process_declarative_part BEGIN
     process_statement_part END PROCESS ';' {result = ProcessDeclaration.new(
     nil, val[0], ); }
  | POSTPONED PROCESS process_declarative_part BEGIN process_statement_part
     END PROCESS ';' {result = ProcessDeclaration.new(nil, val[0], ); }
  | PROCESS_NAME ':' PROCESS process_declarative_part BEGIN
     process_statement_part END PROCESS ';' {result = ProcessDeclaration.new(
     nil, val[0], ); }
  | PROCESS process_declarative_part BEGIN process_statement_part END PROCESS
```

```
    ';' {result = ProcessDeclaration.new(nil, val[0], );; }


process_name :
  PROCESS_NAME {resul}
  identifier {result = val[0]}

process_statement_part :
  process_statement_part_loop0 {result = val}

process_statement_part_loop0 :
  sequential_statement process_statement_part_loop0 {result = val}
  | {result = val}

pure_impure :
  PURE {result = val}
  | IMPURE {result = val}

qualified_expression :
  type_mark '\'' '(' expression ')' {result = val}
  | type_mark '\'' aggregate {result = val}

range_constraint :
  RANGE range_expression {result = val}

range_expression :
  range_attribute_name {result = val}
  | simple_expression direction simple_expression {result = RangeExpression.
     new(val[1], val[0], val[2]);}

record_type_definition :
  RECORD element_declaration record_type_definition_loop0 END RECORD
     record_type_simple_name {result = val}
  | RECORD element_declaration record_type_definition_loop0 END RECORD {result
      = val}

record_type_definition_loop0 :
  element_declaration record_type_definition_loop0 {result = val}
  | {result = val}

relation :
  shift_expression relational_operator shift_expression {result = Relation.new
     (val[0], val[1], val[2]);}
  | function_call relational_operator function_call {result = Relation.new(val
     [0], val[1], val[2]);}
  | shift_expression {result = Relation.new(val[0]);}
  | '('shift_expression ')' {result = Relation.new(val[0]);}




relational_operator :
  '=' {result = val[0]}
```

```
  | '/=' {result = val[0]}
  | '<' {result = val[0]}
  | '<=' {result = val[0]}
  | '>' {result = val[0]}
  | '>=' {result = val[0]}

report_statement :
  LABEL ':' REPORT expression SEVERITY expression ';' {result = val}
  | REPORT expression SEVERITY expression ';' {result = val}
  | LABEL ':' REPORT expression ';' {result = val}
  | REPORT expression ';' {result = val}

return_statement :
  LABEL ':' RETURN expression ';' {result = val}
  | RETURN expression ';' {result = val}
  | LABEL ':' RETURN ';' {result = val}
  | RETURN ';' {result = val}

scalar_type_definition :
  enumeration_type_definition {result = val}
  | integer_type_definition {result = val}
  | floating_type_definition {result = val}
  | physical_type_definition {result = val}

secondary_unit :
  architecture_body {result = val}
  | package_body {result = val}

secondary_unit_declaration :
  identifier '=' physical_literal ';' {result = val}

selected_name :
  prefix '.' suffix {result = val}

selected_signal_assignment :
  WITH expression SELECT target '<=' options selected_waveforms ';' {result =
      val}

selected_waveforms :
  selected_waveforms_loop0 waveform WHEN choices {result = val; }

selected_waveforms_loop0 :
  waveform WHEN choices ',' selected_waveforms_loop0 {result = val;}
  | {result = val;}

sensitivity_clause :
  ON sensitivity_list {result = val}

sensitivity_list :
  identifier_list {result = val[0]}

sensitivity_list_loop0 :
  ',' signal_name sensitivity_list_loop0 {result = val}
```

```
  | ',' label sensitivity_list_loop0 {result = val}

  | {result = val}

sequence_of_statements :
  sequence_of_statements_loop0 {result = val.flatten}

sequence_of_statements_loop0 :
  sequential_statement sequence_of_statements_loop0 {result = val;}
  | {result = val}

sequential_statement :
  wait_statement {result = val; }
  | assertion_statement {result = val; }
  | report_statement {result = val; }
  | signal_assignment_statement {result = val[0]}
  | variable_assignment_statement {result = val[0]; }
  | procedure_call_statement {result = val; }
  | if_statement {result = val[0]; }
  | case_statement {result = val[0]; }
  | loop_statement {result = val[0]; }
  | next_statement {result = val[0]; }
  | exit_statement {result = val[0]; }
  | return_statement {result = val[0]; }
  | null_statement {result = val[0]; }

shift_expression :
  simple_expression shift_operator simple_expression {result = val}
  | simple_expression {result = val[0];}
  | function_call {result = val[0]; }

shift_operator :
  SLL {result = val}
  | SRL {result = val}
  | SLA {result = val}
  | SRA {result = val}
  | ROL {result = val}
  | ROR {result = val}

sign :
  '+' {result = val}
  | '-' {result = val}

signal_assignment_statement :
  LABEL ':' target '<=' delay_mechanism waveform ';' {result =
     SignalAssignmentStatement.new(val[0], val[2], val[4] )}
  | target '<=' delay_mechanism waveform ';' {result =
     SignalAssignmentStatement.new(nil, val[0], val[2])}
  | LABEL ':' target '<=' waveform ';' {result = SignalAssignmentStatement.new
     (val[0],val[2], val[4] )}
  | target '<=' waveform ';' {result = SignalAssignmentStatement.new(nil,val
     [0], val[2] );}
  | target '<=' target ';' {result = SignalAssignmentStatement.new(nil,val[0],
```

```
      val[2] ); ;}
  | function_call '<=' expression ';' {result = SignalAssignmentStatement.new(
      nil,val[0], val[2]); ;}
  | identifier '<=' expression ';' {result = SignalAssignmentStatement.new(nil
      ,val[0], val[2]); ;}
  | target '<=' identifier ';' {result = SignalAssignmentStatement.new(nil,val
      [0], val[2] ); ;}
  | target '<=' function_call ';' {result = SignalAssignmentStatement.new(nil,
      val[0], val[2] );}



signal_declaration :
  SIGNAL identifier_list ':' subtype_indication signal_kind ':=' expression ';
      ' {result = SignalDeclaration.new('SIGNAL', val[1],nil, val[3], val[6]);}
  | SIGNAL identifier_list ':' subtype_indication ':=' expression ';' {result
      = SignalDeclaration.new('SIGNAL', val[1],nil, val[3], val[5]);}
  | SIGNAL identifier_list ':' subtype_indication signal_kind ';' {result =
      SignalDeclaration.new('SIGNAL', val[1],nil, val[3], nil)}
  | SIGNAL identifier_list ':' subtype_indication ';' {result =
      SignalDeclaration.new('SIGNAL', val[1],nil, val[3], nil)}

signal_kind :
  REGISTER {result = val}
  | BUS {result = val}

signal_list :
  signal_name signal_list_loop0 {result = val;}
  | OTHERS {result = val;}
  | ALL {result = val}

signal_list_loop0 :
  ',' signal_name signal_list_loop0 {result = val}
  | {result = val}

signature :
  type_mark signature_loop0 RETURN type_mark {result = val}
  | RETURN type_mark {result = val}
  | type_mark signature_loop0 {result = val}
  | {result = val}

signature_loop0 :
  ',' type_mark signature_loop0 {result = val}
  | {result = val}

simple_expression :
  sign term simple_expression_loop0 {result = val;}
  | term simple_expression_loop0 {result = SimpleExpression.new(nil, val[0],
      val[1]); }
  | '(' term simple_expression_loop0 ')' {result = SimpleExpression.new(nil,
      val[1], val[2], true);}
  | '(' term ')' simple_expression_loop0 {result = SimpleExpression.new(nil,
      val[1], val[2], true);}
```

```
simple_expression_loop0 :
  adding_operator term simple_expression_loop0 {result = val; }
  | {result = val;}

slice_name :
  prefix '(' discrete_range ')' {result = val;}

static_expression :
  expression {result = val[0]}

subprogram_body :
  subprogram_specification IS subprogram_declarative_part BEGIN
     subprogram_statement_part END subprogram_kind designator ';' {result =
     val}
  | subprogram_specification IS subprogram_declarative_part BEGIN
     subprogram_statement_part END designator ';' {result = val}
  | subprogram_specification IS subprogram_declarative_part BEGIN
     subprogram_statement_part END subprogram_kind ';' {result = val}
  | subprogram_specification IS subprogram_declarative_part BEGIN
     subprogram_statement_part END ';' {result = val}

subprogram_declaration :
  subprogram_specification ';' {result = val}

subprogram_declarative_part :
  subprogram_declarative_part_loop0 {result = val}

subprogram_declarative_part_loop0 :
  process_declarative_item subprogram_declarative_part_loop0 {result = val}
  | {result = val}

subprogram_kind :
  PROCEDURE {result = val}
  | FUNCTION {result = val}

subprogram_specification :
  PROCEDURE designator '(' formal_parameter_list ')' {result = val}
  | PROCEDURE designator {result = val}
  | pure_impure FUNCTION designator '(' formal_parameter_list ')' RETURN
     type_mark {result = val}
  | FUNCTION designator '(' formal_parameter_list ')' RETURN type_mark {result
      = val}
  | pure_impure FUNCTION designator RETURN type_mark {result = val}
  | FUNCTION designator RETURN type_mark {result = val}

subprogram_statement_part :
  subprogram_statement_part_loop0 {result = val}

subprogram_statement_part_loop0 :
  sequential_statement subprogram_statement_part_loop0 {result = val}
  | {result = val}
```

```
subtype_declaration :
  SUBTYPE identifier IS subtype_indication ';' {result = val}

subtype_indication :
  resolution_function_name type_mark constraint {result = val}
  | type_mark constraint {result = TypeWrapper.new(val[0], nil, false);}
  | type_mark '(' range_expression ')' {result = TypeWrapper.new(val[0], val
     [2], false); }
  | identifier '(' range_expression ')' {result = TypeWrapper.new(val[0], val
     [2], false);}
  | resolution_function_name type_mark {result = val; }
  | type_mark {result = TypeWrapper.new(val[0], nil, false); }

suffix :
  label {result = val[0]}
  | CHARACTER_LITERAL {result = CharacterLiteral.new(val[0]);
     ConstantValueRepository.add(result);}
  | operator_symbol {result = val[0]}
  | ALL {result = val[0]}

target :
  name {result = val[0]; }
  | aggregate {result = val[0]; }

term :
  factor term_loop0 {result = val.flatten;}
  | '(' factor term_loop0 ')' {result = val.flatten;}

term_loop0 :
  multiplying_operator factor term_loop0 {result = val; binding.pry}
  | {result = nil;}

timeout_clause :
  FOR time_expression {result = val}

type_conversion :
  type_mark '(' expression ')' {result = val}

type_declaration :
  full_type_declaration {result = val}
  | incomplete_type_declaration {result = val}

type_definition :
  scalar_type_definition {result = val}
  | composite_type_definition {result = val}
  | access_type_definition {result = val}
  | file_type_definition {result = val}

type_mark :
  type_name {result = val[0]}
  | subtype_name {result = val[0]}
```

```
type_name :
  name {result = val[0]}

unconstrained_array_definition :
  ARRAY '(' index_subtype_definition unconstrained_array_definition_loop0 ')'
      OF element_subtype_indication {result = val}

unconstrained_array_definition_loop0 :
  ',' index_subtype_definition unconstrained_array_definition_loop0 {result =
      val}
  | {result = val}

use_clause :
  USE selected_name use_clause_loop0 ';' {result = UseClauseWrapper.new(val
      [1,2].flatten.compact);}

use_clause_loop0 :
  ',' selected_name use_clause_loop0 {result = val}
  | {result = val[0]}

variable_assignment_statement :
  LABEL ':' target ':=' expression ';' {result = VariableAssignmentStatement.
      new(val[0], val[2], val[4])}
  | target ':=' expression ';' {result = VariableAssignmentStatement.new(nil,
      val[0], val[2])}

variable_declaration :
  SHARED VARIABLE identifier_list ':' subtype_indication ':=' expression ';' {
      result = val}
  | VARIABLE identifier_list ':' subtype_indication ':=' expression ';' {
      result = val; }
  | SHARED VARIABLE identifier_list ':' subtype_indication ';' {result = val}
  | VARIABLE identifier_list ':' subtype_indication ';' {result = val;}

wait_statement :
  LABEL ':' WAIT sensitivity_clause condition_clause timeout_clause ';' {
      result = val}
  | WAIT sensitivity_clause condition_clause timeout_clause ';' {result = val}
  | LABEL ':' WAIT condition_clause timeout_clause ';' {result = val}
  | WAIT condition_clause timeout_clause ';' {result = val}
  | LABEL ':' WAIT sensitivity_clause timeout_clause ';' {result = val}
  | WAIT sensitivity_clause timeout_clause ';' {result = val}
  | LABEL ':' WAIT timeout_clause ';' {result = val}
  | WAIT timeout_clause ';' {result = val}
  | LABEL ':' WAIT sensitivity_clause condition_clause ';' {result = val}
  | WAIT sensitivity_clause condition_clause ';' {result = val}
  | LABEL ':' WAIT condition_clause ';' {result = val}
  | WAIT condition_clause ';' {result = val}
  | LABEL ':' WAIT sensitivity_clause ';' {result = val}
  | WAIT sensitivity_clause ';' {result = val}
  | LABEL ':' WAIT ';' {result = val}
  | WAIT ';' {result = val}
```

```
waveform :
  waveform_element waveform_loop0 {result = val}
  | UNAFFECTED {result = val}

waveform_element :
  value_expression AFTER time_expression {result = val}
  | value_expression {result = val}
  | identifier {result = val[0];}
  | function_call {result = val[0];}
  | NULL AFTER time_expression {result = val}
  | NULL {result = val}

waveform_loop0 :
  ',' waveform_element waveform_loop0 {result = val}
  | {result = val}


racc_goto_pointer = [
   nil, 155, 159, -1039, -233, 1376, -1152, 1884, -540, -1046,
   nil, 2810, nil, 2023, -92, -108, -59, 94, 2160, -2,
    -9, -5, nil, 159, 240, 55, -8, -204, -106, -252,
  -342, -434, -144, -142, -123, 123, 773, nil, -949, nil,
   nil, -1043, -48, 108, 171, nil, nil, 35, -44, -92,
  -357, nil, nil, -457, -278, -656, -592, -377, nil, -504,
  -648, 74, nil, -39, -105, -36, 78, 224, 86, 306,
   235, nil, 174, 437, 452, -977, -878, -854, -21, -833,
  -406, nil, -1115, -1070, nil, -1025, -1151, -1220, -538, 44,
   -23, -34, -92, nil, -367, -151, -8, nil, -342, nil,
   -12, -9, -295, nil, -341, nil, -525, -501, nil, nil,
  -428, 1190, -104, 73, -774, 731, -845, -487, -931, -724,
   422, 173, nil, -115, 1770, -116, 97, -416, nil, 432,
   nil, nil, nil, nil, nil, 429, 433, -708, 446, 165,
   441, -170, nil, 306, 90, -101, -405, -211, -482, -105,
  -650, 393, 328, 71, nil, -4, -312, nil, 397, 51,
  -468, nil, -202, -474, 42, -650, nil, 128, 130, 133,
   134, nil, -61, 54, -62, -648, 57, -784, -253, nil,
    59, 8, -260, -968, -985, -743, -79, nil, -660, -209,
  -838, 127, nil, -1201, nil, -383, -717, -643, nil, 12,
  -404, 64, nil, nil, nil, nil, -19, nil, -329, -205,
  -480, 487, 489, 490, nil, 145, nil, nil, nil, nil,
   139, nil, nil, nil, -713, nil, 495, 368, nil, 4,
   440, nil, 260, nil, 85, -742, nil, nil, nil, -238,
   799, -238, -645, 883, nil, nil, -593, -624, nil, -564,
   -63, nil, nil, 94, 467, -845, -1300, -754, nil, -450,
   nil, nil, nil, nil, -39, nil, nil, 55, -58, -2,
   266, 136, -109, -456, -237, -551, -17, -802, -831, 194,
  -554, -846 ]

racc_goto_default = [
   nil, nil, nil, 94, nil, nil, 1118, 632, nil, 1122,
  1119, 104, 240, 100, 181, nil, 498, nil, 97, nil,
    82, 35, 103, 101, nil, 61, nil, nil, nil, 1229,
   nil, 1436, nil, nil, nil, 782, nil, 800, 1123, 1121,
```

```
 1128, nil, 499, 300, 591, 39, 32, 34, 500, nil,
  nil, nil, nil, nil, nil, 617, 618, 945, 624, nil,
  nil, 501, 944, 491, 492, 493, 494, 495, 371, 497,
  375, 378, 379, 502, 503, nil, 1228, nil, 1057, 138,
  nil, 597, nil, 1435, 806, 1407, nil, 187, nil, 186,
  321, 183, nil, 946, nil, nil, nil, 598, nil, 469,
  nil, nil, 600, 599, 796, 601, 595, 596, 602, 603,
  802, 98, 214, 84, nil, 812, 899, 900, 901, 916,
  nil, nil, 390, nil, 931, 334, 337, 333, 8, 3,
    4, 5, nil, nil, 403, nil, nil, 1074, 14, nil,
  nil, nil, 225, nil, 81, 863, nil, 715, nil, nil,
  nil, nil, nil, nil, 276, nil, nil, 526, nil, nil,
  nil, 487, nil, 91, nil, nil, 809, 197, 198, 199,
  202, nil, 113, 86, nil, nil, nil, 1126, nil, 176,
  nil, 90, nil, nil, nil, nil, nil, 397, nil, nil,
  nil, 250, 805, nil, 177, nil, nil, nil, 37, nil,
  nil, nil, 402, 399, 400, 401, 405, 409, nil, nil,
  791, nil, nil, nil, 99, nil, nil, 807, nil, 247,
   56, 38, 808, 811, 95, 496, nil, nil, 294, nil,
  nil, 167, nil, nil, nil, nil, 1039, 102, 804, 750,
  nil, 749, nil, nil, nil, nil, 1108, 1109, 173, nil,
  107, 801, 810, nil, nil, nil, nil, nil, nil, 978,
  799, 803, 226, 110, nil, nil, nil, nil, 111, nil,
  277, nil, nil, nil, nil, nil, nil, nil, nil, nil,
  911, nil ]


---- header
require 'strscan'
require 'stringio'
require 'pry'
require_relative 'initializer_repository.rb'
require_relative 'literal_repository.rb'
require_relative 'constant_value_repository.rb'

Dir[File.dirname(__FILE__) + '/wrappers/*.rb'].each {|file| require file}

---- inner
UPPERCASE_LETTERS = 'A-Z'
DIGITS = '0-9'
SPECIAL_CHARS = '\s#&\'\(\)*+,-\.\/\:\;<=>\[\]_|'
SPACE_CHARS = ' '

LOWERCASE_LETTERS = 'a-z'
OTHER_SPECIAL_CHARS = '!$%@?\\^`{}~'
FORMAT_EFFECTOR = '\t\x0B\r\n\x0C'

BASIC_GRAPHIC_CHAR = "#{UPPERCASE_LETTERS}#{DIGITS}#{SPECIAL_CHARS}#{
   SPACE_CHARS}"
GRAPHIC_CHARS = "#{BASIC_GRAPHIC_CHAR}#{LOWERCASE_LETTERS}#{
   OTHER_SPECIAL_CHARS}"
BASIC_CHARS = "#{BASIC_GRAPHIC_CHAR}#{FORMAT_EFFECTOR}"
```

```
BASIC_IDENT_RE = /[A-Za-z][A-Za-z0-9_]*/
EXTENDED_IDENT_RE = /\\([#{GRAPHIC_CHARS}]+)\\/

INTEGER = '[0-9][0-9_]*'
EXPONENT = 'E[+-]?[0-9][0-9_]*'
BASED_INTEGER = '[0-9A-Za-z][0-9A-Za-z_]*'
BIT_VALUE = '[0-9A-Za-z][0-9A-Za-z_]*'

DECIMAL_LITERAL_RE = /#{INTEGER}(?:\.#{INTEGER})?(?:#{EXPONENT})?/
BASED_LITERAL_RE = /#{INTEGER}#(?:#{BASED_INTEGER})(?:\.#{BASED_INTEGER})
   ?#(?:#{EXPONENT})?/
CHAR_LITERAL_RE = /'([#{GRAPHIC_CHARS}])'/
STRING_LITERAL_RE = /"([#{GRAPHIC_CHARS}]*)"/
BIT_STRING_LITERAL_RE = /[BOX]"#{BIT_VALUE}"/
COMMENT_RE = /--.*[\r\n]*/

DELIMITER = '=>|\*\*|:=|\/=|>=|<=|<>|\Z|[&\'()*+,-.\/\:;<=>|\[\]]'
DELIMITER_RE = /(?:#{DELIMITER})/

def parse(str)
 @yydebug = true
 s = StringScanner.new(str)
 def s.collect_token
  tokens = []
  until eos?
   token = yield(self)
   tokens << token if token
  end
  tokens
 end
 @tokens = s.collect_token{|scanner|
  case
  when scanner.skip(/\s+/)
   # ignore
  when scanner.skip(COMMENT_RE)
   # ignore
  when scanner.scan(EXTENDED_IDENT_RE)
   [:EXTENDED_IDENTIFIER, scanner[1]]
  when scanner.scan(BASED_LITERAL_RE)
   [:BASED_LITERAL, scanner[0]]
  when scanner.scan(DECIMAL_LITERAL_RE)
   [:DECIMAL_LITERAL, scanner[0]]
  when scanner.scan(CHAR_LITERAL_RE)
   [:CHARACTER_LITERAL, scanner[1]]
  when scanner.scan(STRING_LITERAL_RE)
   [:STRING_LITERAL, scanner[1]]
  when scanner.scan(BIT_STRING_LITERAL_RE)
   [:BIT_STRING_LITERAL, scanner[0]]
  when scanner.scan(DELIMITER_RE)
   [scanner[0], scanner[0]]
  else
   before = scanner.pos
   scanner.check_until(/\s|#{DELIMITER}/)
```

```
v = scanner.pre_match[before..-1]
scanner.pos += v.size
case v
when /^abs$/i then
  [:ABS, "abs"]
when /^access$/i then
  [:ACCESS, "access"]
when /^after$/i then
  [:AFTER, "after"]
when /^alias$/i then
  [:ALIAS, "alias"]
when /^all$/i then
  [:ALL, "all"]
when /^and$/i then
  [:AND, "and"]
when /^arch$/i then
  [:ARCH, "arch"]
when /^architecture$/i then
  [:ARCHITECTURE, "architecture"]
when /^array$/i then
  [:ARRAY, "array"]
when /^assert$/i then
  [:ASSERT, "assert"]
when /^attr$/i then
  [:ATTR, "attr"]
when /^attribute$/i then
  [:ATTRIBUTE, "attribute"]
when /^begin$/i then
  [:BEGIN, "begin"]
when /^block$/i then
  [:BLOCK, "block"]
when /^body$/i then
  [:BODY, "body"]
when /^buffer$/i then
  [:BUFFER, "buffer"]
when /^bus$/i then
  [:BUS, "bus"]
when /^case$/i then
  [:CASE, "case"]
when /^comp$/i then
  [:COMP, "comp"]
when /^component$/i then
  [:COMPONENT, "component"]
when /^cond$/i then
  [:COND, "cond"]
when /^conditional$/i then
  [:CONDITIONAL, "conditional"]
when /^conf$/i then
  [:CONF, "conf"]
when /^configuration$/i then
  [:CONFIGURATION, "configuration"]
when /^cons$/i then
  [:CONS, "cons"]
```

```
when /^constant$/i then
  [:CONSTANT, "constant"]
when /^disconnect$/i then
  [:DISCONNECT, "disconnect"]
when /^downto$/i then
  [:DOWNTO, "downto"]
when /^else$/i then
  [:ELSE, "else"]
when /^elseif$/i then
  [:ELSEIF, "elseif"]
when /^elsif$/i then
  [:ELSIF, "elsif"]
when /^end$/i then
  [:END, "end"]
when /^entity$/i then
  [:ENTITY, "entity"]
when /^exit$/i then
  [:EXIT, "exit"]
when /^file$/i then
  [:FILE, "file"]
when /^for$/i then
  [:FOR, "for"]
when /^func$/i then
  [:FUNC, "func"]
when /^function$/i then
  [:FUNCTION, "function"]
when /^generic$/i then
  [:GENERIC, "generic"]
when /^generate$/i then
  [:GENERATE, "generate"]
when /^group$/i then
  [:GROUP, "group"]
when /^guarded$/i then
  [:GUARDED, "guarded"]
when /^if$/i then
  [:IF, "if"]
when /^impure$/i then
  [:IMPURE, "impure"]
when /^in$/i then
  [:IN, "in"]
when /^inertial$/i then
  [:INERTIAL, "inertial"]
when /^inout$/i then
  [:INOUT, "inout"]
when /^inst$/i then
  [:INST, "inst"]
when /^instance$/i then
  [:INSTANCE, "instance"]
when /^is$/i then
  [:IS, "is"]
when /^label$/i then
  [:LABEL, "label"]
when /^library$/i then
```

```
  [:LIBRARY, "library"]
when /^linkage$/i then
  [:LINKAGE, "linkage"]
when /^literal$/i then
  [:LITERAL, "literal"]
when /^loop$/i then
  [:LOOP, "loop"]
when /^map$/i then
  [:MAP, "map"]
when /^mod$/i then
  [:MOD, "mod"]
when /^nand$/i then
  [:NAND, "nand"]
when /^new$/i then
  [:NEW, "new"]
when /^next$/i then
  [:NEXT, "next"]
when /^nor$/i then
  [:NOR, "nor"]
when /^not$/i then
  [:NOT, "not"]
when /^null$/i then
  [:NULL, "null"]
when /^of$/i then
  [:OF, "of"]
when /^on$/i then
  [:ON, "on"]
when /^open$/i then
  [:OPEN, "open"]
when /^or$/i then
  [:OR, "or"]
when /^others$/i then
  [:OTHERS, "others"]
when /^out$/i then
  [:OUT, "out"]
when /^pack$/i then
  [:PACK, "pack"]
when /^package$/i then
  [:PACKAGE, "package"]
when /^port$/i then
  [:PORT, "port"]
when /^postponed$/i then
  [:POSTPONED, "postponed"]
when /^procedure$/i then
  [:PROCEDURE, "procedure"]
when /^process$/i then
  [:PROCESS, "process"]
when /^pure$/i then
  [:PURE, "pure"]
when /^range$/i then
  [:RANGE, "range"]
when /^record$/i then
  [:RECORD, "record"]
```

```
when /^register$/i then
  [:REGISTER, "register"]
when /^reject$/i then
  [:REJECT, "reject"]
when /^rem$/i then
  [:REM, "rem"]
when /^report$/i then
  [:REPORT, "report"]
when /^return$/i then
  [:RETURN, "return"]
when /^rol$/i then
  [:ROL, "rol"]
when /^ror$/i then
  [:ROR, "ror"]
when /^select$/i then
  [:SELECT, "select"]
when /^severity$/i then
  [:SEVERITY, "severity"]
when /^shared$/i then
  [:SHARED, "shared"]
when /^sig$/i then
  [:SIG, "sig"]
when /^signal$/i then
  [:SIGNAL, "signal"]
when /^sla$/i then
  [:SLA, "sla"]
when /^sll$/i then
  [:SLL, "sll"]
when /^sra$/i then
  [:SRA, "sra"]
when /^srl$/i then
  [:SRL, "srl"]
when /^subtype$/i then
  [:SUBTYPE, "subtype"]
when /^then$/i then
  [:THEN, "then"]
when /^to$/i then
  [:TO, "to"]
when /^transport$/i then
  [:TRANSPORT, "transport"]
when /^type$/i then
  [:TYPE, "type"]
when /^unaffected$/i then
  [:UNAFFECTED, "unaffected"]
when /^units$/i then
  [:UNITS, "units"]
when /^until$/i then
  [:UNTIL, "until"]
when /^use$/i then
  [:USE, "use"]
when /^var$/i then
  [:VAR, "var"]
when /^variable$/i then
```

```
      [:VARIABLE, "variable"]
    when /^wait$/i then
      [:WAIT, "wait"]
    when /^when$/i then
      [:WHEN, "when"]
    when /^while$/i then
      [:WHILE, "while"]
    when /^with$/i then
      [:WITH, "with"]
    when /^xnor$/i then
      [:XNOR, "xnor"]
    when /^xor$/i then
      [:XOR, "xor"]
    else
      [:BASIC_IDENTIFIER, v]
    end
   end
  }
  @tokens.each_index.select {|i| @tokens[i][0] == :PROCESS}.each do |
     process_index|
   if @tokens[process_index-2][0] == :BASIC_IDENTIFIER && @tokens[
     process_index-1][0] == ':'
    @tokens[process_index-2][0] = :PROCESS_NAME
   end
  end
  do_parse
end

def next_token
  @tokens.shift
end

---- footer
VERSION = '0.3.2';
parser = VHDLParser::new
begin
  if ARGV.length == 0
   usage = <<DOCOPT
    Vhdl obfuscator v.#{VERSION}
    Eugene Shadura. Belarussian State University of Informatics and
       Radioelectronics

    Usage:
    ruby #{__FILE__} -i <path to input file>...
    ruby #{__FILE__} -i <path to input file> -a
    ruby #{__FILE__} -i <path to input file> -o <path to output file>
    ruby #{__FILE__} -i <path to input file> [-lexical-only | --functional-
       only]
    ruby #{__FILE__} -h | help
    ruby #{__FILE__} version

    Options:
    -h help Show this screen.
```

```
    version Show version.
    -i select input file
    -a analyze code correctness without obfuscation
    -o print result to file
    -lexical-only obfuscate literal names only without adding complexity to
       RTL level
    --functional-only obfuscale RTL level synthesis by adding primitives
       without changing literal names

DOCOPT
   puts usage
  else
   if ARGV.include? '-i'
    if ARGV.include? '-a'
     puts "Starting code checking..."
    end
    $lexical_obfuscation_mode = true
    ast = parser.parse(open(ARGV[1]).read)
    unless ARGV.include? '-a'
     puts ast
    else
     ast[0].statements.select{|el| el.class.name=="EntityDeclaration"}.each
        do |entity|
      puts "Code check for entity '#{entity.name.name}' completed."
     end
     puts "Code correct"
    end
   end
  end
rescue ParseError => e
puts $!
end
```