Санкт–Петербургский государственный университет

Царёв Никита Евгеньевич

Выпускная квалификационная работа

Разработка обучающего веб-инструмента удаленной сборки и интерактивной отладки программ

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2018 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Современное программирование»

Научный руководитель:
профессор, факультет математики и компьютерных наук Санкт–Петербургского государственного университета, д.ф. - м.н. Куликов Александр Сергеевич

Рецензент:
доцент Новгородского государственного университета имени Ярослава Мудрого, к. т. н. Довгалюк Павел Михайлович

Санкт-Петербург
2022 г.

Содержание

Введение

В настоящее время обучение языку ассемблера является важной составляющей многих программистских курсов.

Очень часто студенты, изучающие язык ассемблера, сталкиваются с проблемами при настройки среды разработки, при использовании инструментов компиляции и отладки.

Преподаватели таких курсов также сталкиваются с проблемами организации учебного процесса.

Создание удобного, интерактивного и производительного программного инструмента удалённой сборки и отладки програм на ассемблере, представляет собой актуальную задачу.

Постановка задачи

Цель данной работы состоит в разработке обучающего веб-инструмента удалённого запуска, отладки и проверки программ на языке ассемблера.

Задачи данной работы:

1. Исследование существующих решений для запуска и отладки программ на языке ассемблера, а также решений для обучения языку ассемблера.
2. Формирование требований к разрабатываемому инструменту.
3. Исследование возможности создания инструмента.
4. Разработка программной архитектуры инструмента.
5. Реализация инструмента.
6. Исследование свойств решения.

Объектом исследования являются системы запуска и отладки программ на языке ассемблера.

Предметом исследования является интерактивность и удобство использования таких систем.

Практическая ценность работы состоит в том, что разработанный инструмент позволит проводить обучение языку ассемблера более эффективно для студентов.

Глава 1. Обзор предметной области

1.1 Критерии сравнения

В мире существует множество решений для запуска ассемблерного кода, а также решений для обучения языку ассемблера. В этой главе рассматривается несколько таких решений, каждое из них анализируется в контексте следующих **критериев сравнения**:

1. Поддержка запуска ассемблерного кода на разных диалектах и на разных архитектурах.
2. Поддержка отладки: выполнение по шагам, поддержка точек останова, редактирования регистров/памяти, визуализация стека вызовов.
3. Поддержка задач и их автоматической проверки.
4. Поддержка интеграции с системами управления обучением.
5. Возможность работы без установки дополнительного программного обеспечения на устройстве пользователя.
6. Возможность самостоятельной установки и развёртывания системы на выделенном сервере, доступность исходного кода.

1.2 Существующие решения

1.2.1 Ideone

Ideone[1] является онлайн компилятором и средой разработки, поддерживающей более 60 языков программирования, в том числе несколько диалектов ассемблера.

[1] `https://ideone.com`

Поддерживается запуск ассемблерного кода на архитектурах x86 (NASM и GNU диалекты) и x86-64 (только NASM диалект). Отладка не поддерживается.

Поддержки задач, их автоматической проверки нет, соответственно нет и интеграции в системы управления обучением.

Взаимодействие с системой происходит через веб-интерфейс, установки дополнительного ПО не требуется.

Система имеет закрытый исходный код, самостоятельно установить систему на выделенный сервер не представляется возможным.

1.2.2 OneCompiler

OneCompiler[2] является онлайн компилятором и средой разработки, поддерживающей, в том числе, и NASM диалект x86 ассемблера.

Поддержиывается запуск кода, есть возможность указать содержимое стандартного потока ввода перед запуском. Отладка не поддерживается.

Поддержки задач, их автоматической проверки нет, соответственно нет и интеграции в системы управления обучением.

Взаимодействие с системой происходит через веб-интерфейс, установки дополнительного ПО не требуется.

Система имеет закрытый исходный код, самостоятельно установить систему на выделенный сервер не представляется возможным.

1.2.3 ASM Debugger

ASM Debugger[3] является инструментом для пошаговой отладки простых программ на языке ассемблера.

Особенностью инструмента является то, что он не использует запуск программ на реальном аппаратном обеспечении. Вместо этого, на языке Javascript реализовано подмножество инструкций x86 ассемблера.

Поддерживается запуск ассемблерного кода на архитектуре x86 с NASM

[2] `https://onecompiler.com/assembly`

[3] `http://asmdebugger.com`

диалектом. Поддерживается пошаговое исполнение, просмотр значений регистров.

Поддержки задач, их автоматической проверки нет, соответственно нет и интеграции в системы управления обучением.

Взаимодействие с инструменетом происходит через веб-интерфейс, установки допольнительного ПО не требуется.

Инструмент имеет открытый исходный код[4], соответственно есть возможность установить его на выделенный сервер.

1.2.4 SASM (SimpleASM)

SASM[5] представляет из себя кроссплатформенную среду разработки на языке ассемблера для архитектур x86 и x86-64 с использованием диалектов NASM, GNU, FASM, MASM.

Поддерживается запуск ассемблерного кода, поддерживается выполнение по шагам, точки останова, просмотр и редактирование регистров и памяти, а также произвольные команды GDB.

Поддержки задач, их автоматической проверки нет, соответственно нет и интеграции в системы управления обучением.

Для использования инструмента необходима его установка на компьютер пользователя. Инструмент имеет открытый исходный код[6].

1.2.5 JetBrains Clion + EduTools

Clion[7] — это интегрированная среда разработки от компании JetBrains, предназначенная, в первую очередь, для разработки приложений на языках C и C++. Язык ассемблера не поддерживается ни в каком виде, но существуют сторонние плагины, которые решают эту проблему, например NASM Assembly Language[8].

[4] https://github.com/dinoqqq/asmdebugger

[5] https://dman95.github.io/SASM/index.html

[6] https://github.com/Dman95/SASM

[7] https://www.jetbrains.com/clion/

[8] https://plugins.jetbrains.com/plugin/9759-nasm-assembly-language

Компиляция и запуск кода на языке ассемблера возможны, если модифицировать должным образом файлы системы описания сборки CMake. Отладка ассемблерного кода не поддерживается.

Плагин EduTools[9] позволяет создавать и писать задачи с автоматическими тестами, что упрощает проверку решений. Отсутствует поддержка задач с закрытыми (недоступными для обучающегося) тестами. Интеграция с системами управления обучением отсутствует.

Для использования данной среды разработки необходима её установка на компьютер пользователя. Она имеет закрытый исходный код.

1.2.6 GitHub Classroom + Visual Studio Code

GitHub Classroom — это сервис, позволяющий давать учебные задания в виде git-репозиториев. GitHub Classroom позволяет добавить кнопку «открыть в Visual Studio Code», которая позволяет открыть репозиторий с преднастроенными плагинами в этом редакторе.

Для того, чтобы настроить поддержку языка ассемблера в Visual Studio Code, требуется установка дополнительных плагинов. Также преподавателю в шаблонном репозитории необходимо будет настроить компиляцию и запуск в файлах `tasks.json` и `launch.json`. Отладка не поддерживается.

GitHub Classroom позволяет добавлять тесты через веб-интерфейс преподавателя. В качестве теста может выступать набор входных данных и эталонных ответов к ним, так и путь до скрипта для автоматической проверки. В первом случае входные данные передаются программе через стандартный поток ввода, а вывод программы сравнивается с эталонным ответом.

Необходима установка Visual Studio Code, компилятора и отладчика.

GitHub Classroom имеет закрытый исходный код, установить свою копию на выделенный сервер не представляется возможным.

[9] `https://plugins.jetbrains.com/plugin/10081-edutools`

1.2.7 Stepik

В системе управления обучением Stepik[10] есть режим задания Code Challenge, который позволяет проверять код, написанный на различных языках программирования.

Поддерживается NASM диалект x86 и x86-64 ассемблера. Отладка не поддерживается.

Поддерживаются задачи и их автоматическая проверка на скрытых тестах. Тесты должны иметь вид набора входных данных и эталонных ответов. Входные данные передаются программе через стандартный поток ввода, а вывод программы сравнивается с эталонным ответом.

Взаимодействие с системой происходит через веб-интерфейс, установка дополнительного ПО не требуется. Система имеет закрытый исходный код.

1.2.8 Moodle + Virtual Programming Lab

Для системы управления обучением Moodle[11] существует плагин Virtual Programming Lab[12], котторый позволяет запускать и проверять код, написанный на различных языках программирования.

Поддерживается NASM диалект x86 ассемблера, отладка не поддерживается.

Поддерживаются задачи и их автоматическая проверка на скрытых тестах. Тесты должны иметь вид набора входных данных и эталонных ответов. Входные данные передаются программе через стандартный поток ввода, а вывод программы сравнивается с эталонным ответом.

Взаимодействие с системой происходит через веб-интерфейс, установка дополнительного ПО не требуется. И система Moodle и плагин Virtual Programming Lab имеют открытый исходный код. Соответственно, есть возможность установки этой связки на выделенный сервер.

[10] `https://stepik.org`

[11] `https://moodle.org/`

[12] `https://vpl.dis.ulpgc.es`

1.2.9 Git репозиторий с заданиями и скриптами для проверки

Одним из популярных способов организовать проверку заданий при проведении курсов по программированию является специально организованный git репозиторий. К примеру может ожидаться, что студенты выполняют задания в своих локальных копиях, а задания проверяются специально написанными скриптами, расположенными в том же репозитории.

При таком подходе, учащиеся могут выбирать любой удобный им редактор кода. Для запуска кода могут быть предоставлены готовые скрипты. То же самое нельзя сказать об отладке, скорее всего студенту придётся познакомиться с gdb или другой подобной утилитой.

Такой подход применяется в учебном процессе на практике. Возможно автоматизировать и генерацию репозиториев с задачами для каждого пользователя, и закрытые тесты, и даже интеграцию с системами управления обучением. На практике таким мало кто занимается.

Такой подход требует от орагнизаторов курса реализовать скрипты запуска и проверки решений. От студентов же требуется установка дополнительного программного обеспечения (такого как git, компилятор, отладчик) и наличие навыков работы с инструментами командной строки. Также может требоваться использование конкретной операционной системы, так как зачастую скрипты расчитывают на наличие конкретного пользовательского окружения.

1.3 Сравнение существующих решений

Краткая информация о решениях представлена в таблице 1.

Таблица 1: Сравнение существующих решений

Решение	**Диалект**	**Отладка**	**Учебный процесс**	**Нужна установка**	**Открытый исходный код**

Ideone	x86 (NASM, GNU), x86_64 (NASM)	нет	нет	нет	нет
OneCompiler	x86 NASM	нет	нет	нет	нет
ASM Debugger	x86 NASM (подмножество)	да[13]	нет	нет	да
SASM	x86, x86_64 (NASM, GNU, FASM, MASM)	да	нет	да	да
Clion + EduTools	x86 NASM[14]	нет	частично[15]	да	нет
GitHub + VSCode	зависит от настроек	нет	частично	да	нет
Stepik	x86, x86_64 NASM	нет	да	нет	нет
Moodle + VPL	x86 NASM	нет	да	нет	да
Git репозиторий	зависит	нет	частично	да	—

Среди расмотренных альтернатив можно выделить несколько групп схожих решений.

Первой такой группой являются онлайн компиляторы. К ним можно отнести Ideone и OneCompiler. Они представляют веб редакторы с возможностью компиляции и запуска кода на разных языках. В этих системах языку ассемблера не уделяется особого внимания, так как основная масса пользователей таких систем использует их для написания кода на высокоуровневых языках. Также эти системы имеют закрытый исходный код.

[13]Поддерживается пошаговое исполнение, просмотр значений регистров

[14]Только со сторонним плагином

[15]Задачи поддерживаются, интеграции с системами управления обучением нет

В качестве второй группы можно выделить такие системы как Stepik, Moodle и JetBrains EduTools. Они предназначены, в первую очередь, для образовательных процессов. Поддержка задач, направленных на иизучение языка ассемблера, не является их основной целью. Так, для системы Moodle, требуется сторонний плагин, а среды разработки JetBrains поддерживают механизм отладки многих языков программирования, но не ассемблера.

Также хочется выделить решения, требующие сложной настройки со стороны преподавателей, такие как интеграция GitHub Classroom с Visual Studio Code и использование git репозитория для организации учебного процесса. Эти решения отличаются гибкостью, но требуют написания скриптов и конфигурационных файлов перед использованием. Также они требуют установки дополнительного программного обеспечения на компьютерах студентов.

В качестве последней категории можно выделить средства, предназначенные именно для запуска и отладки ассемблерного кода, но не поддерживающие интеграцию в учебный процесс, такие как ASM Debugger и SASM. Так как эти инструменты направлены именно на работу с языком ассемблера, они обладают большим функционалом по сравнению с другими, более общими решениями. Например, они поддерживают отладку программ на ассемблере, чего не поддерживает ни одна другая рассмотренная альтернатива.

1.4 Выводы

В этой главе были рассмотрены различные решения для запуска и отладки программ на языке ассемблера, а также решения, предназначенные для обучения языку ассемблера. Исходя из обзора предметной области, можно сделать вывод, что имеется недостаток решений, одновременно позволяющих производить отладку программ на языке ассемблера, интегрироваться в учебный процесс и не требовать установки и настройки на устройстве пользователя.

Глава 2. Формулировка требований к решению

2.1 Функциональные требования

Наиболее важными характеристиками разрабатываемого инструмента будут те, которые выгодно его отличают от существующих аналогов. Исходя из этого, были сформулированы следующие функциональные требования к инструменту:

1. Инструмент должен быть доступен через веб-интерфейс и не требовать установки дополнительного программного обеспечения на устройстве пользователя.

2. Инструмент должен содержать возможность аутентификации пользователей, как по паре логин/пароль, так и через протокол LTI.

3. Должна существовать возможность разделения прав пользователей на администраторов и учащихся.

4. Администраторы должны иметь возможность создавать, редактировать и удалять задачи.

5. Каждая задача должна иметь: название, текст условия, связанный с ней чекер и его параметры.

6. Учащиеся должны уметь получать задания по конкретным задачам через системы управления обучением. Для этого администратор такой системы должен настроить LTI интеграцию, в частности, указать идентификатор нужной задачи.

7. На странице задания для учащегося должно быть доступно условие задачи, редактор кода, возможность отправить решение на проверку и информация о предыдущих попытках решения.

8. Вместе с редактором кода должна быть доступна функциональность отладки: добавление и удаление точек останова, запуск, остановка, пошаговое исполнение программы. Если программа находится в приоста-

новленном состоянии, должен быть доступен просмотр и редактирование регистров процессора и содержимого памяти процесса. Должна поддерживаться архитектура x86_64.

9. Запускаемые пользовательские программы должны быть ограничены по времени и используемой памяти, им должен быть запрещен доступ к файловой системе, сети, процессам и другим ресурсам операционной системы.

2.2 Нефункциональные требования

1. Инструмент, в первую очередь, предназначается для задач, направленных непосредственно на изучение языка ассемблера. Таким образом, задачи должны быть составлены так, чтобы их решения могли исполняться в контексте непривилегированных процессов пользовательского пространства, без доступа к конкретным системным вызовам и периферии.

2. Инструмент должен минимизировать количество элементарных шагов, требуемых для запуска программ.

3. Инструмент должен быть эффективен по использованию процессорного времени и оперативной памяти.

4. Инструмент должен обладать достаточным быстродействием и отзывчивостью.

Глава 3. Архитектура программной реализации

3.1 Структура программной реализации

Реализация инструмента представляет из себя несколько сервисов, общающихся между собой по различным протоколам. Схема взаимодействия процессов в системе представлена на рисунке 1.

Сервис **web** представляет из себя приложение, написанное на языке Python с использованием фреймворка Flask. Он отвечает за основную бизнес-

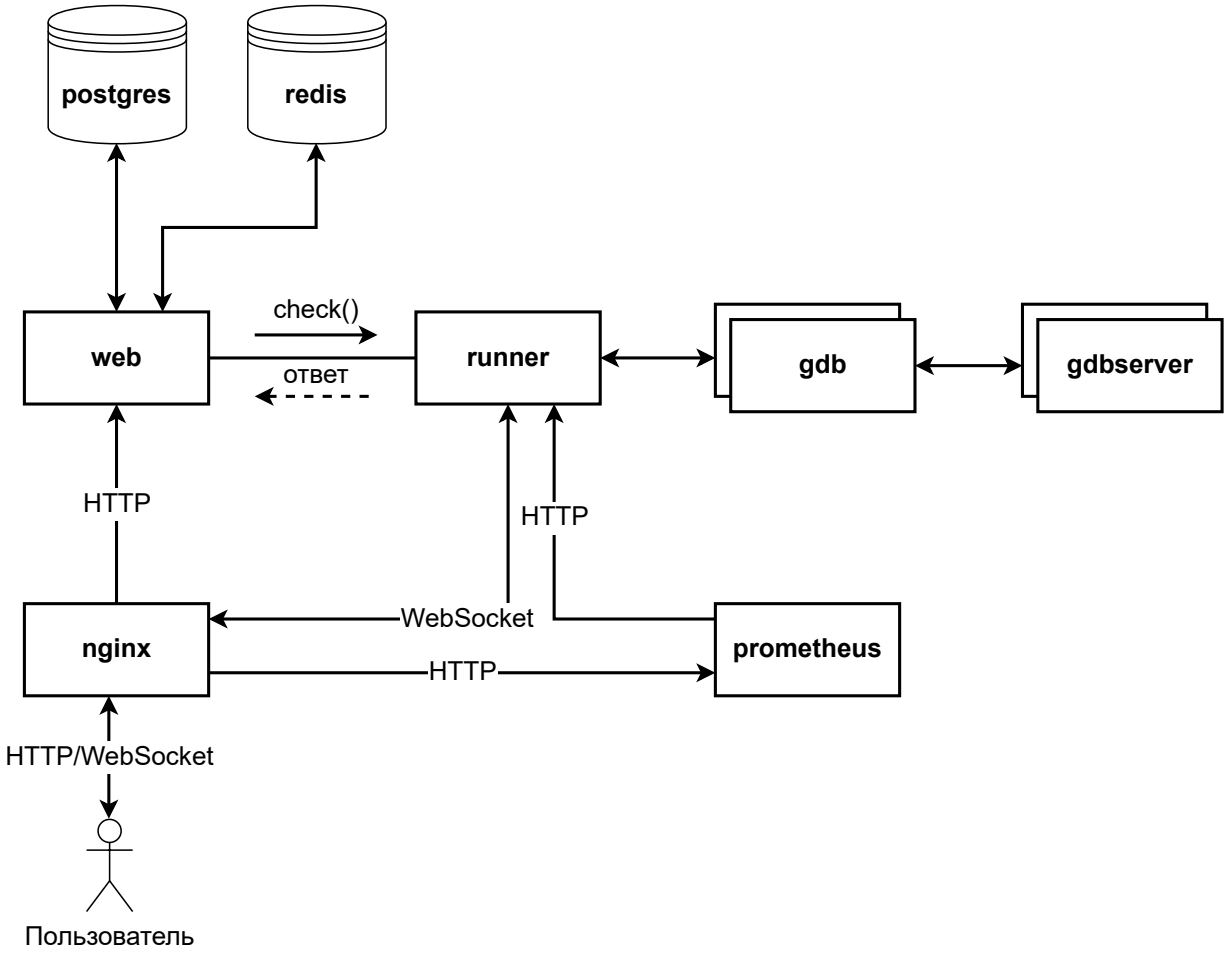

Рис. 1: Схема взаимодействия процессов в системе

логику инструмента. Сервис предоставляет функционал аутентификации и авторизации, как через протокол LTI, так и по паре логин/пароль. Для студентов сервис предоставляет интерфейс просмотра задач и решений, а для преподавателей — панель управления, позволяющую создавать и редактировать задачи, просматривать попытки решения и информацию о пользователях.

Сервис **runner** представляет из себя приложение, написанное на языке Python с использованием библиотеки AIOHTTP. Он отвечает за управление сессиями отладки, взаимодействие с процессами отладчика через протокол GDB/MI, а также за взаимодействие с пользователями через протокол WebSocket. Помимо предоставления возможности интерактивной отладки веб-интерфейсу, этот сервис также занимается автоматизированной проверкой решений. Эта функциональность недоступна для внешнего пользователя напрямую, запросы на проверку решений отправляет сервис web по протоколу

HTTP.

Базы данных **postgres** и **redis** используются для хранения необходимой для работы системы информации. В PostgreSQL хранится информация о пользователях, задачах, заданиях и посылках. Также там хранится метаинформация об интерактивных сессиях отладки. В Redis хранится множество использованных значений nonce при авторизации систем управления обучением по протоколу LTI.

Система мониторинга **prometheus** предоставляет интерфейс для просмотра различных метрик, описывающих состояние системы. Метрики собираются как и с самого Prometheus, так и с сервиса runner через протокол HTTP.

Веб-сервер **nginx** используется в качестве обратного прокси-сервера, проксирующего HTTP запросы в сервис web и взаимодействие по протоколу WebSocket с сервисом runner. Также Nginx отдаёт статические файлы и проксирует сервис мониторинга Prometheus. Перед доступом к Prometheus, Nginx подтверждает у web наличие у пользователя прав администратора.

3.2 Компиляция ассемблерных программ

Используем GCC. Поддерживаются директивы `#line`, таким образом можно чинить номера строк с отладочной информации и сообщениях об ошибках.

3.3 Запуск ассемблерных программ

3.3.1 Изоляция с помощью seccomp

Seccomp — механизм ядра Linux, позволяющий процессу перейти в «безопасный режим», в котором запрещены все системные вызовы, кроме `exit`, `sigreturn`, `read` и `write`. Запретить работать со стандартными потоками ввода/вывода можно, предварительно вызвав `close` на них.

Здесь можно привести листинг кода, который переводит программу в безопасный режим. Код этот живёт в файле `environment/x86_64/entry.S`.

3.3.2 Ограничение потребляемых ресурсов с помощью Docker

Docker (через механизм контрольных групп) позволяет ограничивать использование процессорного времени, используемую память в контейнерах.

Существует системный вызов `setrlimit`. Ограничиваем процессорное время в секундах через `RLIM_CPU`. Docker умеет это делать через параметр `ulimit`.

3.4 Отладка ассемблерных программ

3.4.1 Использование GDB

GDB — консольный инструмент отладки программ. Позволяет отлаживать программы на самых разных языках программирования на разных платформах. Поддерживает отладочную информацию в формате DWARF.

3.4.2 GDB/MI

GDB/MI (GDB Machine Interface, машинный интерфейс GDB) позволяет взаимодействовать с процессом отладки в машиночитаемом виде. Это нам пригодится.

Тут можно вкратце описать формат взаимодействия, сослаться на мануал GDB/MI.

3.4.3 GDB server

GDB server — программа, с которой GDB может взаимодействовать, чтобы организовать отладку кода на удалённой машине. Пригодится для разделения полномочий и ограничения ресурсов.

3.5 Взаимодействие с пользователем через веб-браузер

Для интерактивной отладки необходимо не только передавать команды из веб-интерфейса в GDB, но и асинхронно реагировать на события, возникающие при отладки. К таким событиям, например, относится остановка

программы на точке останова. К счастью, все современные браузеры поддерживают протокол WebSocket, который позволяет общаться клиенту и серверу полностью асинхронно, а не по модели запрос-ответ.

3.6 Взаимодействие с системами управления обучением

Существует такой протокол: LTI, Learning Tools Interoperability. В частности, позволяет по протоколу OAuth авторизовывать пользователей конкретной LMS, получать информацию о таске, а также отправлять результаты проверки как score от 0.0 до 1.0.

3.7 Сбор метрик

Поднимаем Prometheus, пишем туда нужные метрики. Идеи для релевантных метрик: задержка исполнения команд gdb, время реакции на команды пользователя, общее потребление памяти на процесс отладки (gdb + gdbserver + программа), потребление cpu на процесс отладки (интересует idle cpu usage).

3.8 Интерфейс пользователя

3.8.1 Аутентификация через логин и пароль

Есть у нас страница `/login`. Ожидается, что так входят только админы.

3.8.2 Аутентификация через LTI

При правильно настроенном Moodle, достаточно зайти на страницу задания и увидеть iframe с интерфейсом пользователя, вход происходит автоматически. Под капотом нам POST запрос к ручке `/lti` с нужными параметрами.

Скриншот мудла с iframeом.

3.8.3 Работа с интерактивным отладчиком через GUI

Можно ставить брейкпоинты, можно слать команды, можно смотреть на регистры. Обратите вниманеи на красивый редактор. Скриншот в paused состоянии.

3.8.4 Работа с админской панелью

Есть несколько сущностей: пользователи, задачи, задания и посылки, с ними работа ведётся одинаково.

3.9 Модель данных

3.9.1 Схема базы данных

Тут схема, которую как-то надо нарисовать.

3.9.2 Таблица users

Содержит в себе данные о пользователях, в том числе и метод аутентификации (password или LTI).

3.9.3 Таблица problems

Содержит информацию о задачах, в том числе и чекер.

3.9.4 Таблица assignments

Задания, выданные студентам. Содержит, помимо прочего, LTI callback и оценку.

3.9.5 Таблица submissions

Содержит посылки по задачам.

Глава 4. Исследование свойств решения

Выводы

Ну вот написали инструмент, все задачи поставленные во введении сделали.

Список литературы