

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт Информационных технологий

Кафедра корпоративных ИС

Практическая работа №2

По дисциплине: «Конфигурационное управление»

**Студент группы** ИКБО-12-22 Белоклоков Роман Валентинович

Москва, 2023

# Задание

Написать на выбранном вами языке программирования программу, которая принимает в качестве аргумента командной строки имя пакета, а возвращает граф его зависимостей в виде текста на языке Graphviz. На выбор: для npm или для pip. Пользоваться самими этими менеджерами пакетов запрещено. Главное, чтобы программа работала даже с неустановленными пакетами и без использования pip/npm.

## Исходный код

Листинг  - Исходный код файла pyproject.toml

|  |
| --- |
| [tool.poetry]  name = "dependency‑visualizer"  version = "0.1.0"  description = ""  authors = ["witelokk <death.thunder@yandex.ru>"]  readme = "README.md"  [tool.poetry.dependencies]  python = "^3.11"  requests = "^2.31.0"  [build‑system]  requires = ["poetry‑core"]  build‑backend = "poetry.core.masonry.api" |

Листинг  - Исходный код файла dependency\_visualizer/\_\_main\_\_.py

|  |
| --- |
| from argparse import ArgumentParser  from .dependency\_visualizer import NpmDependencyVisualizer  def **main**():  arg\_parser = ArgumentParser("dependency\_visualizer")  arg\_parser.add\_argument("package\_name")  parsed\_args = arg\_parser.parse\_args()  dv = NpmDependencyVisualizer(parsed\_args.package\_name)  print(dv.get\_graph())  if **\_\_name\_\_** == "\_\_main\_\_":  main() |

Листинг  - Исходный код файла dependency\_visualizer/dependency\_visualizer.py

|  |
| --- |
| import requests  import json  class NpmDependencyVisualizer:  def **\_\_init\_\_**(self, package\_name: str) ‑> **None**:  self.\_package\_name = package\_name  def **generate\_dependencies**(self, package\_name: str) ‑> list[str]:  r = requests.get(f"https://registry.npmjs.org/{package\_name}")  response = json.loads(r.content)  try:  latest\_version = response["dist‑tags"]["latest"]  return list(response["versions"][latest\_version]  ["dependencies"].keys())  except Exception:  return []  def **get\_graph**(self) ‑> str:  """Returns a Graphviz graph of dependencies"""  xs = [self.\_package\_name]  graph = f'"{self.\_package\_name}";'  while xs:  x = xs.pop()  dependencies = self.generate\_dependencies(x)  for dependency in dependencies:  if '"'+dependency+'"' not in graph:  xs.append(dependency)  graph += f'"{x}" ‑> "{dependency}";'  return f"digraph {{ {graph} }}" |

## Примеры

Примеры графов, сгенерированных программой представлены на рисунках 1, 2.

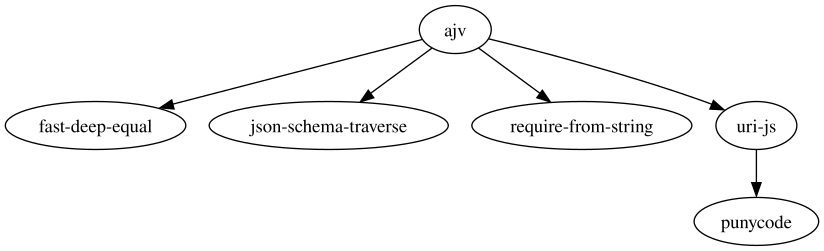


Рисунок  - Граф зависимостей пакета ajv

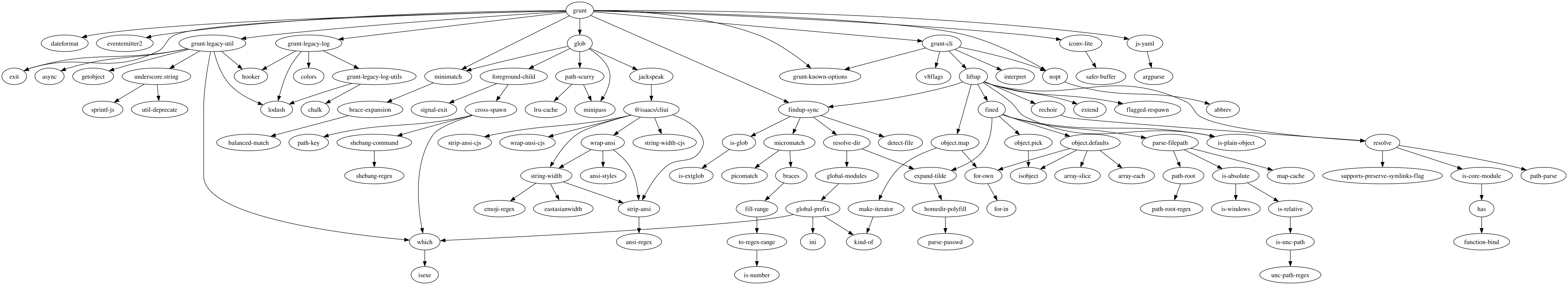


Рисунок  - Граф зависимостей пакета grunt

## Вывод

При выполнения данной работы были изучены работа различных пакетных менеджеров. В результате был написана программа для генерации графов зависимостей npm‑пакетов для Graphviz.