

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

**DEVOPS: ЕГО РОЛЬ В СОВРЕМЕННОЙ РАЗРАБОТКЕ
ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

РЕФЕРАТ

студента 3 курса 351 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Устюшина Богдана Антоновича

Проверено:

доцент, к. п. н.

А. П. Грецова

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Основные задачи DevOps	5
1.1 Ускорение разработки, её инструменты достижения	5
1.1.1 CI и CD	5
1.1.2 Основные средства автоматизации	5
1.1.3 Git	6
1.1.4 Docker	7
1.1.5 Kubernetes	9
1.1.6 Возможные перспективы	10
2 Безопасность и мониторинг в DevOps	11
2.1 Безопасность	11
2.2 Мониторинг	12
3 Какими навыками должен обладать DevOps инженер?	14
4 Вакансии и финансовые перспективы	16
ЗАКЛЮЧЕНИЕ	17
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	18
Приложение А Скриншоты с сайта вакансий	19

ВВЕДЕНИЕ

DevOps – это культура и набор практик, направленных на сближение и сотрудничество между разработчиками и операционными инженерами с целью автоматизации процессов разработки, тестирования, доставки и управления инфраструктурой. Главная цель DevOps – улучшить скорость и надежность разработки и эксплуатации программного обеспечения. Развитие культуры и практик DevOps имеет огромное значение в современной сфере разработки программного обеспечения по нескольким ключевым причинам:

1. Ускорение разработки и поставки. DevOps позволяет автоматизировать процессы сборки, тестирования и развертывания приложений, что уменьшает время, необходимое для поставки новых функций и исправлений.
2. Повышение надежности и стабильности. Благодаря автоматизации и стандартизации процессов, DevOps сокращает количество ошибок, связанных с развертыванием и конфигурацией инфраструктуры, что повышает надежность системы.
3. Эффективное использование ресурсов. Автоматизация и оптимизация процессов управления ресурсами позволяют более эффективно использовать вычислительные мощности и инфраструктуру.
4. Улучшение коммуникации и сотрудничества. DevOps способствует созданию единой команды, объединяющей разработчиков и операционных инженеров, что улучшает коммуникацию и сотрудничество внутри организации.
5. Безопасность. DevOps позволяет внедрять безопасность с самого начала разработки, включая в себя практики DevSecOps для раннего выявления и предотвращения уязвимостей.
6. Масштабируемость и гибкость. DevOps позволяет эффективно масштабировать инфраструктуру в зависимости от потребностей бизнеса и адаптироваться к изменяющимся условиям.
7. Повышение качества продукта. Благодаря автоматизации тестирования и развертывания, DevOps способствует повышению качества программного обеспечения.
8. Экономия времени и ресурсов. Автоматизация процессов сокращает время, затраченное на рутинные операции, и позволяет сотрудникам сосредотачиваться на более стратегических задачах.

9. Улучшение удовлетворенности клиентов. Быстрая поставка новых функций и низкая вероятность сбоев влияют на удовлетворенность клиентов продуктом.

1 Основные задачи DevOps

1.1 Ускорение разработки, её инструменты достижения

1.1.1 CI и CD

Одними из наиболее употребляемых терминов в задачах DevOps являются CI/CD.

CI расшифровывается как **Continuous Integration** (непрерывная интеграция). Это практика в разработке программного обеспечения, которая заключается в автоматическом объединении всех изменений кода от разных участников команды в общий репозиторий после каждого коммита.

Цель непрерывной интеграции – обеспечить регулярное и автоматизированное тестирование нового кода в общей среде, чтобы своевременно выявлять и решать конфликты, ошибки и проблемы, которые могут возникнуть из-за взаимодействия различных частей программы. Это также способствует поддержанию высокого качества кода и улучшению сотрудничества в команде разработчиков.

CD в контексте DevOps расшифровывается как **Continuous Delivery** (непрерывная поставка). Это практика автоматизированной подготовки программного продукта к выкладыванию в продакшн среду после каждого успешного завершения процесса CI. В Continuous Delivery приложение всегда готово к релизу, но решение о том, когда релизнуть, принимает человек.

Continuous Deployment (непрерывное развертывание): Это практика автоматического развертывания каждой успешной сборки в продакшн среду без необходимости вмешательства человека. Система принимает решение о выкладывании новой версии автоматически.

Иногда оба эти термина объединяются под аббревиатурой CI/CD, чтобы описать полный цикл непрерывной разработки и поставки программного обеспечения [1].

1.1.2 Основные средства автоматизации

Рассмотрим основные средства автоматизации процессов DevOps инженера в современной разработке:

- Git и системы контроля версий
- Docker
- Kubernetes

- Ansible, Puppet, Chef
- Terraform
- Prometheus и Grafana
- Jenkins
- ELK Stack (Elasticsearch, Logstash, Kibana)
- Системы управления конфигурациями облачных платформ (например, AWS CloudFormation, Google Cloud Deployment Manager)
- Системы мониторинга и управления логами (например, Splunk, Sumo Logic)

Опишем наиболее важные из них.

1.1.3 Git

Система управления версиями (также используется определение “система контроля версий”, от англ. *version control system*) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое [2].

Git является одной из самых популярных систем контроля версий [3]. Git играет важную роль в практиках DevOps, особенно в контексте управления исходным кодом и совместной работы над проектами. Вот несколько способов, как Git поддерживает DevOps:

1. Управление версиями кода. Git позволяет разработчикам эффективно управлять версиями своего кода, фиксировать изменения и возвращаться к предыдущим состояниям проекта.
2. Ветвление и слияние. Git обеспечивает мощные инструменты для создания веток (branches) в репозитории, что позволяет параллельно работать над разными функциональностями или исправлениями без влияния на основную ветку.
3. Коллаборативная разработка. Git обеспечивает возможность множеству разработчиков работать над одним проектом, сливая свои изменения в общий репозиторий.
4. Интеграция с CI/CD. Git интегрируется с инструментами непрерывной интеграции и поставки (CI/CD), такими как Jenkins, GitLab CI/CD, и другими. Это позволяет автоматически запускать сборки и разворачивать приложе-

ния при обновлении репозитория.

5. Ревью кода и Pull Requests. Git-платформы (например, GitHub, GitLab, Bitbucket) предоставляют функционал для обзора кода и создания запросов на включение (Pull Requests), что улучшает качество кода и содействует сотрудничеству.
6. Отслеживание изменений и анализ кода. Git позволяет отслеживать и анализировать изменения в коде, что помогает в обнаружении и решении проблем на ранних этапах.
7. Резервное копирование и восстановление. Git-репозитории предоставляют механизмы для резервного копирования кода, что обеспечивает сохранность данных и возможность восстановления в случае необходимости.
8. Управление конфликтами и версиями данных. Git позволяет эффективно управлять конфликтами при слиянии изменений и версиями данных в проекте.

Git является неотъемлемой частью DevOps практик, обеспечивая эффективное управление кодовой базой, совместную работу разработчиков и автоматизацию процессов разработки и поставки.

1.1.4 Docker

Прежде чем говорить про Docker, нужно сказать несколько слов о технологии контейнеризации.

Контейнеры — это способ стандартизации развертки приложения и отделения его от общей инфраструктуры. Экземпляр приложения запускается в изолированной среде, не влияющей на основную операционную систему.

Разработчикам не нужно задумываться, в каком окружении будет работать их приложение, будут ли там нужные настройки и зависимости. Они просто создают приложение, упаковывают все зависимости и настройки в некоторый единый образ. Затем этот образ можно запускать на других системах, не беспокоясь, что приложение не запустится.

Docker – это открытая платформа для разработки, доставки и эксплуатации приложений. Docker разработан для более быстрого выкладывания ваших приложений. С помощью Docker вы можете отделить ваше приложение от вашей инфраструктуры и обращаться с инфраструктурой как управляемым приложением. Docker помогает выкладывать ваш код быстрее, быстрее тестировать,

быстрее выкладывать приложения и уменьшить время между написанием кода и запуска кода. Docker позволяет создавать контейнеры, автоматизировать их запуск и развертывание, управляет жизненным циклом. Он позволяет запускать множество контейнеров на одной хост-машине. [4] [5].

Вот несколько способов, как Docker поддерживает DevOps:

1. Унификация среды разработки и продакшн. Docker контейнеры позволяют упаковывать приложения и их зависимости в изолированные среды, что обеспечивает согласованность между средами разработки и продакшн.
2. Повышение эффективности разработчиков. Разработчики могут создавать контейнеры с необходимым окружением и зависимостями, что упрощает процесс настройки и развертывания приложений.
3. Быстрая развертка и масштабирование. Docker обеспечивает быстрое создание и запуск контейнеров, а также их масштабирование в зависимости от нагрузки.
4. Управление конфигурацией как кодом. Docker-файлы (Dockerfile) позволяют описывать конфигурацию контейнера в виде кода, что упрощает процесс создания и управления контейнерами.
5. Интеграция с CI/CD. Docker интегрируется с инструментами непрерывной интеграции и поставки (CI/CD), что позволяет автоматически собирать и развертывать контейнеры при обновлении кодовой базы.
6. Изоляция и безопасность. Docker контейнеры обеспечивают изоляцию приложений, что предотвращает вмешательство в работу других процессов и улучшает безопасность.
7. Управление ресурсами. Docker позволяет точно настраивать выделение ресурсов для контейнеров, что позволяет эффективно использовать аппаратное обеспечение.
8. Легкость миграции и обновления. Docker позволяет быстро переносить и обновлять приложения, что упрощает управление и поддержку приложений в продакшн среде.
9. Совместимость с оркестраторами контейнеров. Docker совместим с оркестраторами, такими как Kubernetes, что позволяет управлять крупными и сложными инфраструктурами контейнеров.

Использование Docker в DevOps помогает создавать надежные, масштабируемые и эффективные инфраструктуры для разработки, тестирования и раз-

вертывания приложений.

1.1.5 Kubernetes

Kubernetes – это портативная расширяемая платформа с открытым исходным кодом для управления контейнеризованными рабочими нагрузками и сервисами, которая облегчает как декларативную настройку, так и автоматизацию. У платформы есть большая, быстро растущая экосистема. Сервисы, поддержка и инструменты Kubernetes широко доступны [6].

Kubernetes играет важную роль в DevOps практиках, особенно в управлении контейнеризованными приложениями и оркестрации их развертывания. Вот несколько способов, как Kubernetes поддерживает DevOps:

1. Оркестрация контейнеров. Kubernetes предоставляет мощные средства для управления контейнерами, автоматизируя процессы развертывания, масштабирования и управления жизненным циклом приложений.
2. Автоматическое масштабирование. Kubernetes позволяет автоматически масштабировать приложения в зависимости от текущей нагрузки, что обеспечивает эффективное использование ресурсов.
3. Декларативное управление. С помощью YAML-файлов, Kubernetes позволяет описывать желаемое состояние приложения, а оркестратор самостоятельно следит за его выполнением.
4. Высокая доступность и надежность. Kubernetes предоставляет механизмы для обеспечения непрерывной работы приложений, включая управление репликациями, обнаружение отказов и автоматическое восстановление.
5. Разделение ресурсов и изоляция. Kubernetes обеспечивает возможность точной настройки выделения ресурсов для контейнеров и изоляции их друг от друга.
6. Обновление приложений без простоев. Kubernetes позволяет проводить обновления приложений с минимальным воздействием на работу системы.
7. Использование нейтральных к облаку ресурсов. Kubernetes абстрагирует вычислительные, сетевые и хранилищеские ресурсы, что позволяет работать с различными облачными и локальными провайдерами.
8. Интеграция с CI/CD. Kubernetes интегрируется с инструментами непрерывной интеграции и поставки (CI/CD), что позволяет автоматически развертывать приложения в кластере.
9. Мониторинг и управление состоянием. Kubernetes предоставляет сред-

ства для мониторинга состояния приложений и кластера, а также для реагирования на изменения.

10. Расширяемость и гибкость. Kubernetes предоставляет API и плагиновую архитектуру, что позволяет расширять его функциональность и адаптировать к специфическим потребностям.

Использование Kubernetes в DevOps практиках позволяет управлять сложными инфраструктурами контейнеров, обеспечивая высокую доступность, масштабируемость и надежность приложений.

1.1.6 Возможные перспективы

В связи с современным ростом популярности искусственного интеллекта мы можем наблюдать видоизменение профессии программного инженера (в том числе и DevOps-инженера). Так, уже существуют вакансии так называемых “Prompt-engineers” – инженера-подсказчика. Он специализируется на работе с AI (например, ChatGPT) и умеет создавать запросы для обучения и/или грамотно составлять запросы с целью выполнения ИИ поставленной задачи [7]. Развитие LLM-моделей, в том числе и ChatGPT определённо изменит подход к разработке ПО уже в ближайшее время.

В заключение обзора инструментов хочется отметить, что все они помогают автоматизировать различные аспекты процессов разработки, тестирования, развертывания и управления инфраструктурой в рамках DevOps методологии.

2 Безопасность и мониторинг в DevOps

2.1 Безопасность

Безопасность в DevOps – это важный аспект, который охватывает практики и принципы обеспечения защиты и безопасности при проектировании, разработке, развертывании и эксплуатации приложений и инфраструктуры. Вот некоторые ключевые аспекты безопасности в DevOps:

1. Интеграция безопасности с начала. Безопасность должна быть встроена в каждый этап жизненного цикла разработки, начиная с проектирования и архитектуры приложения.
2. Автоматизированные тесты безопасности. В DevOps следует включать автоматизированные тесты безопасности в непрерывные пайплайны для выявления и устранения уязвимостей на ранних этапах разработки.
3. Контроль доступа и привилегий. Необходимо строго управлять доступом к ресурсам и системам, предоставляя минимальные необходимые привилегии.
4. Мониторинг и аудит безопасности. Реализация мониторинга событий и аудита для быстрого обнаружения и реагирования на потенциальные инциденты безопасности.
5. Управление уязвимостями и обновления. Регулярно сканировать инфраструктуру и приложения на предмет уязвимостей и обновлять компоненты для устранения обнаруженных проблем.
6. Шифрование и защита данных. Использовать шифрование данных в покое и в передаче, а также обеспечивать безопасное хранение паролей и ключей.
7. Безопасность кода и сборок. Внедрять проверки безопасности в код и проверять код на предмет известных уязвимостей перед развертыванием.
8. Системы обнаружения вторжений (IDS) и предотвращения вторжений (IPS). Использовать IDS/IPS для обнаружения и предотвращения атак на инфраструктуру и приложения.
9. Безопасность контейнеров и оркестрации. Обеспечивать безопасность контейнеров с использованием проверенных образов, а также применять рекомендации безопасности для оркестраторов, таких как Kubernetes.
10. Обучение и осведомленность сотрудников. Обучать членов команды DevOps основам безопасности и поддерживать их в курсе текущих угроз и лучших практик.

11. Резервное копирование и восстановление данных. Регулярно создавать резервные копии данных и проверять процессы восстановления для гарантированного восстановления в случае инцидента.

Обеспечение безопасности в DevOps является ключевым элементом успешного функционирования приложений и инфраструктуры, и оно должно быть в центре всего процесса разработки и эксплуатации [8].

2.2 Мониторинг

Мониторинг в контексте DevOps – это процесс непрерывного сбора, анализа и отображения данных о работе приложений, инфраструктуры и процессов разработки и развертывания. Он направлен на обеспечение надежности, производительности и доступности системы в реальном времени.

Мониторинг в DevOps включает в себя сбор данных с различных источников (логи, метрики, события), их анализ, а также предоставление информации в понятной и удобной форме для принятия оперативных и стратегических решений [9].

Основные принципы мониторинга в DevOps включают в себя следующие аспекты:

1. Непрерывность (Continuous Monitoring). Мониторинг должен быть непрерывным процессом, который работает в реальном времени, обеспечивая постоянный контроль за состоянием приложений и инфраструктуры.
2. Автоматизация (Automation). Мониторинг должен быть автоматизирован для быстрого обнаружения и реагирования на аномалии или проблемы.
3. Централизованность (Centralization). Сбор и анализ данных из различных источников (серверов, приложений, сетей и т.д.) должны быть централизованы для удобства анализа и отслеживания.
4. Измеримость (Measurability). Метрики и показатели, собранные в ходе мониторинга, должны быть измеряемыми и иметь смысл для оценки производительности и состояния системы.
5. Оповещения и уведомления (Alerting and Notification). Мониторинг должен предоставлять механизмы для оповещения о выявленных проблемах, позволяя оперативно реагировать на них.
6. Специфичность (Specificity). Мониторинг должен быть настроен на отслеживание конкретных аспектов приложений и инфраструктуры, соответствующих целям и требованиям бизнеса.

7. Реакция на аномалии (Anomaly Response). Мониторинг должен быть способен автоматически выявлять аномалии в работе приложений и системы и предпринимать соответствующие действия для их решения.
 8. Отчетность (Reporting). Мониторинг должен предоставлять средства для создания отчетов и анализа данных для обеспечения прозрачности и инсайтов в работу системы.
 9. Масштабируемость (Scalability). Мониторинг должен быть способен масштабироваться в соответствии с ростом размеров и сложности инфраструктуры и приложений.
 10. Резервирование и восстановление (Backup and Recovery). Должны быть предусмотрены механизмы резервирования данных и настроек мониторинга для обеспечения возможности быстрого восстановления в случае сбоев.
 11. Сбор данных и аналитика (Data Collection and Analytics). Мониторинг должен предоставлять средства сбора, хранения и анализа данных для выявления трендов и понимания долгосрочных изменений в системе.
- Эти принципы помогают обеспечить эффективный и надежный мониторинг приложений и инфраструктуры в рамках DevOps практик.

3 Какими навыками должен обладать DevOps инженер?

Конечно, основные навыки, которые должен иметь DevOps-инженер, схожи с теми, что требуются для других разработчиков. Однако следует рассмотреть их с целью понимания того, чем занимается DevOps.

Помимо технических навыков, описанных выше, у DevOps-инженера должны быть определенные личностные качества, которые помогают ему эффективно выполнять свою работу. Вот некоторые из них:

1. **Ответственность.** DevOps инженер должен быть ответственным за свою работу и за результаты своих действий. Он должен понимать важность своей роли в разработке и поддержке инфраструктуры.
2. **Ориентированность на результат.** Эффективный DevOps инженер стремится к достижению поставленных целей и выполнению задач в срок. Он умеет работать в условиях давления и соблюдать приоритеты.
3. **Гибкость и адаптивность.** Работа в DevOps требует способности быстро адаптироваться к изменяющимся обстоятельствам и новым технологиям. Инженер должен быть гибким в подходе к решению задач.
4. **Способность к сотрудничеству.** DevOps инженер часто работает в команде с разработчиками, тестировщиками, системными администраторами и другими специалистами. Умение эффективно коммуницировать и сотрудничать с коллегами – важное качество.
5. **Проактивность.** Хороший DevOps инженер стремится к постоянному совершенствованию и предпринимает инициативу в решении проблем и оптимизации процессов.
6. **Стрессоустойчивость.** Работа в DevOps может включать в себя срочные ситуации и неожиданные проблемы. Инженер должен уметь сохранять спокойствие и принимать решения в условиях стресса.
7. **Аналитическое мышление.** DevOps инженер должен быть способным анализировать сложные ситуации, выявлять причины проблем и разрабатывать эффективные решения.
8. **Лидерство и инициативность.** В некоторых ситуациях DevOps инженер может выступать в роли лидера команды или инициатора изменений в процессах и инфраструктуре.
9. **Постоянное обучение.** Быстро меняющаяся технологическая среда требует от DevOps инженера стремления к постоянному обучению и освоению

новых инструментов и методов.

10. Внимание к деталям. При работе с инфраструктурой и конфигурациями, важно быть внимательным к деталям, чтобы избегать потенциальных проблем.

Эти личностные качества помогают DevOps инженеру эффективно выполнять свою работу и успешно взаимодействовать с командой разработки и операций.

4 Вакансии и финансовые перспективы

На сегодняшний день (ноябрь 2023 года) в Москве около 2800 вакансий по запросу DevOps инженер (был использован наиболее популярный в России сайт по поиску работы hh.ru) 1. Достаточно высокие и зарплаты, отсутствует так называемый “потолок” 2. Около трети вакансий на сайте предполагают удалённую работу или гибкий график, что также удобно 3.

В Саратове на сегодняшний день есть 19 вакансий DevOps-инженера, что показывает не очень большую заинтересованность в наличии отдельного DevOps в небольших компаниях и филиалах больших компаний. Это связано с тем, что DevOps в большей степени оптимизирует нагрузку, появляющуюся в результате большой загруженности как со стороны программистов, так и со стороны пользователей. Малый бизнес, которого в нестоличном регионе больше, меньше нуждается в подобной оптимизации.

Тем не менее, при анализе статистики вакансий столичного региона мы можем судить о хорошей финансовой и рыночной перспективе становления DevOps инженером.

ЗАКЛЮЧЕНИЕ

В этом реферате были рассмотрены основные инструменты DevOps инженера.

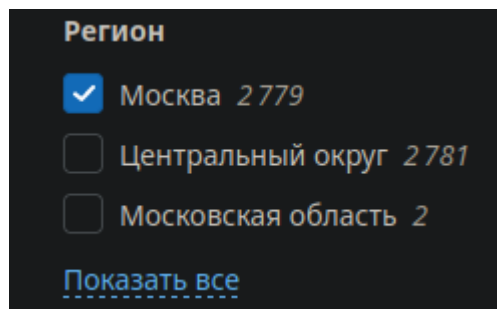
Конечно, владеть этими технологиями может не только отдельно взятый человек, но и все члены команды разработки. Сегодня распространена тенденция нанимать людей (особенно в большие компании на высокие должности), которые знают основы администрирования, умеют пользоваться вышеописанными инструментами и в целом обладают некоторыми наборами практик DevOps. Тем не менее, отдельно взятый инженер – это невероятно востребованная профессия на рынке разработки ПО. Все эти аспекты делают DevOps неотъемлемой частью современной разработки ПО, особенно в компаниях, где важны высокая скорость разработки и надежность продукта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Что такое ci/cd? Разбираемся с непрерывной интеграцией и непрерывной поставкой. — URL: <https://habr.com/ru/companies/otus/articles/515078/>.
- 2 Distributed version control systems: A not-so-quick guide through. — URL: <https://www.infoq.com/articles/dvcs-guide/>.
- 3 Статистика популярности систем контроля версий. — URL: <https://strawpoll.com/most-popular-version-control-system>.
- 4 Что такое docker: для чего он нужен и где используется. — URL: <https://selectel.ru/blog/what-is-docker> (Дата обращения 12.07.2013).
- 5 Понимая docker. — URL: <https://habr.com/ru/articles/253877/>.
- 6 Что такое kubernetes? — URL: <https://kubernetes.io/ru/docs/concepts/overview/what-is-kubernetes/>.
- 7 Вакансия prompt-инженера. — URL: <https://uk.indeed.com/viewjob?jk=7bd6b6bd58342cad&from=serp&vjs=3>.
- 8 Что такое безопасность devops? — URL: https://www.keepersecurity.com/ru_RU/resources/glossary/what-is-devops-security/.
- 9 Инструменты для мониторинга архитектуры. — URL: <https://proglab.io/p/12-instrumentov-devops-inzhenera-dlya-monitoringa-arhitektury-2020-02-13>.

ПРИЛОЖЕНИЕ А

Скриншоты с сайта вакансий

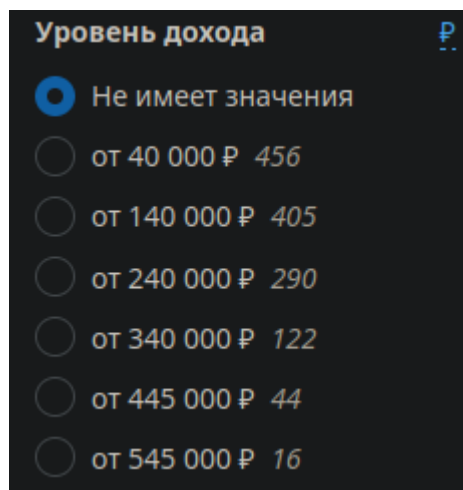


Регион

- ☒ Москва 2 779
- ☐ Центральный округ 2 781
- ☐ Московская область 2

[Показать все](#)

Рисунок 1 – Количество вакансий в Москве



Уровень дохода ₽

- ☒ Не имеет значения
- ☐ от 40 000 ₽ 456
- ☐ от 140 000 ₽ 405
- ☐ от 240 000 ₽ 290
- ☐ от 340 000 ₽ 122
- ☐ от 445 000 ₽ 44
- ☐ от 545 000 ₽ 16

Рисунок 2 – Уровень дохода в Москве(из 2779 вакансий не у всех указан)

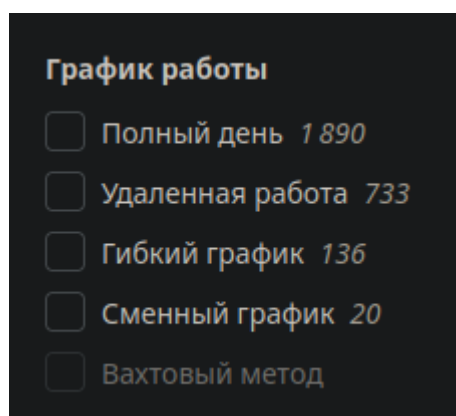


График работы

- ☐ Полный день 1 890
- ☐ Удаленная работа 733
- ☐ Гибкий график 136
- ☐ Сменный график 20
- ☐ Вахтовый метод

Рисунок 3 – График работы в Москве