СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc160829490)

[1 АРХИТЕКТУРА LTE 6](#_Toc160829491)

[2 DDD И SOLID 10](#_Toc160829492)

[3 ПРАКТИЧЕСКАЯ ЧАСТЬ 13](#_Toc160829493)

[ЗАКЛЮЧЕНИЕ 19](#_Toc160829494)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 20](#_Toc160829495)

# ВВЕДЕНИЕ

Стандарт 4G, или четвёртое поколение сетей мобильной связи, представляет собой значительный шаг вперёд в сравнении с предыдущими технологиями. Он обеспечивает более высокую скорость передачи данных, улучшенную пропускную способность, низкую задержку (латентность) и повышенную эффективность сети. 4G был разработан для удовлетворения растущего спроса на беспроводную передачу данных и обеспечения более качественного опыта использования мобильного интернета.

Одним из наиболее широко распространенных стандартов 4G является Long-Term Evolution (LTE). LTE представляет собой технологию, которая обеспечивает высокоскоростную передачу данных, начиная с частоты 100 Мбит/с и более высокой, с возможностью увеличения скорости до нескольких сотен мегабит в секунду в идеальных условиях.

Преимущества стандарта 4G включают в себя:

Высокая скорость передачи данных: Стандарт 4G обеспечивает значительно более высокие скорости передачи данных по сравнению с предыдущими стандартами, что позволяет пользователям быстро загружать и скачивать контент из интернета.

Улучшенная пропускная способность: 4G обеспечивает более широкие каналы передачи данных, что позволяет более эффективно использовать доступные ресурсы сети.

Низкая задержка: LTE обеспечивает низкую задержку данных, что делает его идеальным для приложений, требующих быстрого отклика, таких как онлайн-игры и видеозвонки.

Поддержка множества устройств: Стандарт 4G поддерживает большое количество одновременных подключений, что позволяет обслуживать большое количество пользователей одновременно.

Однако, несмотря на свои многочисленные преимущества, стандарт 4G также имеет свои ограничения, такие как ограниченная покрытие в отдаленных районах и высокие затраты на инфраструктуру. Тем не менее, 4G остается ключевым компонентом современных мобильных сетей и является основой для развертывания последующих поколений сетей мобильной связи, таких как 5G.

В настоящее время технологии мобильной связи играют ключевую роль в повседневной жизни людей, обеспечивая им доступ к широкому спектру информации и услуг в любой точке мира. Одним из самых важных достижений в области мобильной связи является внедрение технологии Long-Term Evolution (LTE), которая представляет собой стандарт беспроводной связи следующего поколения.

LTE представляет собой стандарт высокоскоростной беспроводной передачи данных, который обеспечивает значительно более высокие скорости передачи данных по сравнению с предыдущими технологиями. Эта технология позволяет пользователям наслаждаться быстрым доступом в интернет, стримингом видео высокого разрешения, онлайн-играми и другими передовыми приложениями, требующими высокой пропускной способности сети.

В данной курсовой работе мы проведем обзор технологии LTE, рассмотрим её ключевые характеристики, принцип работы, преимущества и недостатки, а также её роль в современной мобильной связи. Также мы рассмотрим перспективы развития LTE и его влияние на будущее беспроводных коммуникаций.

Сравнение стандарта 4G (LTE) с 3G имеет несколько ключевых аспектов, которые отражают существенное различие между этими двумя поколениями мобильной связи:

* Скорость передачи данных: Одним из наиболее заметных различий между 4G и 3G является скорость передачи данных. Стандарт 4G обеспечивает гораздо более высокие скорости, чем 3G. В то время как 3G может достигать скоростей до нескольких мегабит в секунду (Mbps), 4G обычно обеспечивает скорости в несколько десятков мегабит в секунду (Mbps) и даже более высокие.
* Качество передачи данных: 4G предоставляет более высокое качество передачи данных благодаря использованию передовых технологий, таких как Orthogonal Frequency Division Multiplexing (OFDM) и Multiple Input Multiple Output (MIMO), которые обеспечивают более эффективное использование радиочастотного спектра и улучшают устойчивость сигнала. Это делает 4G более надежным для приложений, требующих высокой пропускной способности.
* Задержка (латентность): Стандарт 4G также обеспечивает более низкую задержку (или латентность) по сравнению с 3G. Это означает, что приложения, требующие быстрой реакции, такие как онлайн-игры или видеозвонки, работают более плавно и без задержек.
* Эффективность использования спектра: 4G использует более эффективные методы управления радио ресурсами, что позволяет обеспечивать более высокую пропускную способность при том же спектральном диапазоне по сравнению с 3G. Это позволяет сетям 4G обеспечивать более высокий объем данных на одну единицу спектра, что является критически важным в условиях растущего спроса на мобильные данные.
* Поддержка голосовой связи: В отличие от 3G, где голосовая связь осуществляется как обычный поток данных, 4G использует технологии, такие как Voice over LTE (VoLTE), для передачи голоса в цифровом формате, что обеспечивает более высокое качество звука и лучшее управление голосовыми вызовами.
* Применение и возможности: 4G открывает двери для различных инновационных приложений и сервисов, таких как мобильное видео высокого разрешения, потоковая передача музыки, облачные вычисления и многие другие, которые требуют высокой скорости и стабильного подключения.

# АРХИТЕКТУРА LTE

Изображение выглядит как диаграмма, зарисовка, Технический чертеж, План

Автоматически созданное описание

Рисунок 1.1 – Схема LTE

Архитектурные узлы: Архитектура LTE состоит из нескольких ключевых узлов, включая Evolved Node B (eNodeB), Mobility Management Entity (MME), Serving Gateway (SGW), Packet Data Network Gateway (PDN-GW), PCRF (Policy and Charging Rules Function), TA (Tracking Area) и Home Subscriber Server (HSS). Каждый узел выполняет определенные функции, такие как обработка данных, управление мобильностью, маршрутизация трафика и аутентификация пользователей.

eNodeB (Evolved Node B): это базовая станция LTE, которая обеспечивает беспроводное соединение с мобильными устройствами. eNodeB отвечает за передачу данных, управление сигнализацией и управление ресурсами радиоканала.

MME (Mobility Management Entity): MME отвечает за управление мобильностью пользователей в сети LTE. Он обрабатывает процессы регистрации, аутентификации и авторизации пользователей, а также управляет перемещениями между различными узлами сети.

SGW (Serving Gateway): SGW выполняет функцию маршрутизации и пересылки пакетов данных между eNodeB и другими сетями, такими как Internet или другие мобильные сети. Он также отвечает за управление мобильностью внутри сети LTE.

PDN-GW (Packet Data Network Gateway): PDN-GW предоставляет доступ к пакетным сетям данных, таким как Internet или корпоративные сети. Он выполняет функции маршрутизации данных, NAT (Network Address Translation) и обеспечивает соединение между мобильной сетью LTE и внешними сетями.

HSS (Home Subscriber Server): HSS является центральным узлом для управления подписчиками в LTE-сети. Он хранит информацию об абонентских данных, таких как аутентификационные параметры, профили услуг и прочее.

PCRF (Policy and Charging Rules Function): является центральным элементом управления политиками и правилами для обеспечения качества обслуживания и управления тарификацией в LTE-сети. Он определяет политики для управления потоками данных в сети, такие как приоритет трафика, ограничения скорости передачи данных, доступ к определенным службам и приложениям. PCRF также отвечает за динамическое применение тарифных планов и тарификацию услуг в зависимости от требований сети и пользователей.

TA (Tracking Area): представляет собой группу сот LTE, которые могут быть обслужены одним и тем же MME (Mobility Management Entity) в сети. Концепция TA используется для уменьшения нагрузки на MME и оптимизации процесса мобильности в LTE-сети. Вся мобильная сеть LTE разделена на регионы, каждый из которых состоит из нескольких TAs. Когда мобильное устройство перемещается между TA, оно сообщает об изменении своего местоположения MME, чтобы обновить информацию о его текущем TA. При переходе между TA мобильное устройство может изменить TA только с одной TA на другую, избегая необходимости обновления MME. Это сокращает сигнализацию в сети и улучшает производительность мобильности.Начало формы

Интерфейсы: В архитектуре LTE существует несколько интерфейсов между различными узлами, такие как интерфейс S1 между eNodeB и MME, интерфейс S5/S8 между MME и SGW, интерфейс S11 между MME и SGW, и интерфейс S6a между MME и HSS. Эти интерфейсы обеспечивают обмен данных и сигнализации между узлами сети LTE.

Развитие сети: LTE является эволюцией предыдущих технологий сотовой связи и обеспечивает путь для развития к более продвинутым технологиям, таким как LTE-Advanced и LTE-Advanced Pro, которые предлагают улучшенные скорости передачи данных, увеличенную емкость сети и другие дополнительные возможности.

Порядок установления соединения:

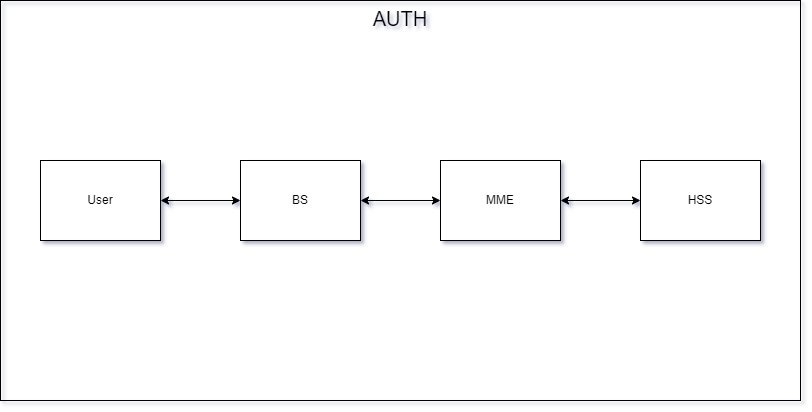


Рисунок 1.2 – Порядок авторизации пользователя

Авторизация происходит по пути, как показано на рисунке 1.2. Изначально пользователь отправляет запрос на базовую станцию. Далее идет перенаправление в MME, где происходит процесс аутентификации пользователя с HSS. После этого пользователь получает информацию о SGW, с которой нужно установить связь.

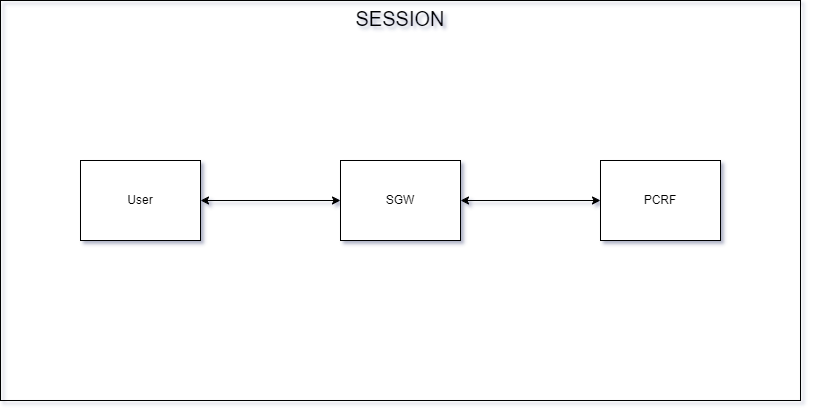


Рисунок 1.3 – Порядок получения сессии

Получение сессии для пользователя происходит по схеме, изображенной на рисунке 1.3. В этом случае, чтобы получить сессию нужно обратиться к SGW, полученному на прошлом шаге. SGW в свою очередь делегирует эту задачу PCRF, где относительно клиента выбираются параметры сессии.

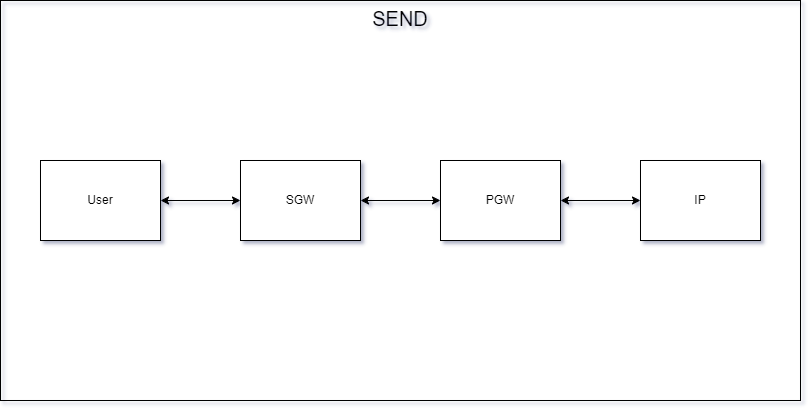


Рисунок 1.4 – Порядок отправки сообщения

На рисунке 1.4. изображена схема отправки сообщений. Для этого клиенту все также нужно из-за его уровня абстракции обращаться к SGW. Можно заметить, что название Serving Gateway – обслуживающий шлюз, как раз он выступает прокси относительно клиента и всей оставшейся сети. То есть при помощи него клиент отделен от конкретики сети. По ту сторону SGW может быть любой вид сети. PGW выступает как связь с другими сетями LTE.

# DDD И SOLID

Не сложно заметить, но прошлая схема имеет достаточно весомый недостаток. Она не подходит под сети другого поколения. Поэтому не сложно заметить некоторые абстракции, к которым можно прийти и которые в будущем могут облегчить запоминание схем сетей сотовой связи, ведь они будут иметь одну и ту же абстрактную схему и все различия будут заключаться лишь в конкретике.

Суть DDD заключается в том, что наружная часть домена ничего не знает о внутренней части домена и она всего лишь делегирует ей задачу, пришедшую к ней снаружи, то есть этот слой представляет собой слой связи. Все что находится ниже этого слоя, представляет уже из себя логику системы, и она не зависит от того, кто прислал то или иное задание на обработку. Это похоже на дизайн паттерн – Адаптер, который представляет из себя слой “Адаптации”, чтобы внутренняя логика не была подвержена изменению из вне.

Не трудно заметить, что принципы DDD очень напоминают принципы – SOLID:

* Принцип единственной ответственности (Single Responsibility Principle - SRP): Каждый класс должен иметь только одну причину для изменения. Это означает, что класс должен быть ответственным только за одну часть функциональности программы. Если класс выполняет более одной задачи, его лучше разделить на несколько классов.
* Принцип открытости/закрытости (Open/Closed Principle - OCP): Программные сущности (классы, модули, функции) должны быть открыты для расширения, но закрыты для изменения. Это означает, что поведение сущности должно быть изменяемым без изменения самой сущности. Это достигается, например, с помощью использования полиморфизма и абстракций.
* Принцип подстановки Барбары Лисков (Liskov Substitution Principle - LSP): Объекты в программе должны быть заменяемыми на экземпляры их подтипов без изменения корректности программы. Другими словами, подтип должен дополнять, а не изменять базовый тип.
* Принцип разделения интерфейса (Interface Segregation Principle - ISP): Клиенты не должны зависеть от интерфейсов, которые они не используют. Более специфичные интерфейсы предпочтительнее, чем один общий интерфейс. Это помогает избежать наличия в коде зависимостей от ненужных методов и свойств.
* Принцип инверсии зависимостей (Dependency Inversion Principle - DIP): Модули верхнего уровня не должны зависеть от модулей нижнего уровня. Оба типа модулей должны зависеть от абстракций. Инверсия зависимостей также предполагает, что абстракции не должны зависеть от деталей реализации, а детали реализации должны зависеть от абстракций.

Абстрактная архитектура любой сотовой сети будет выглядеть так:

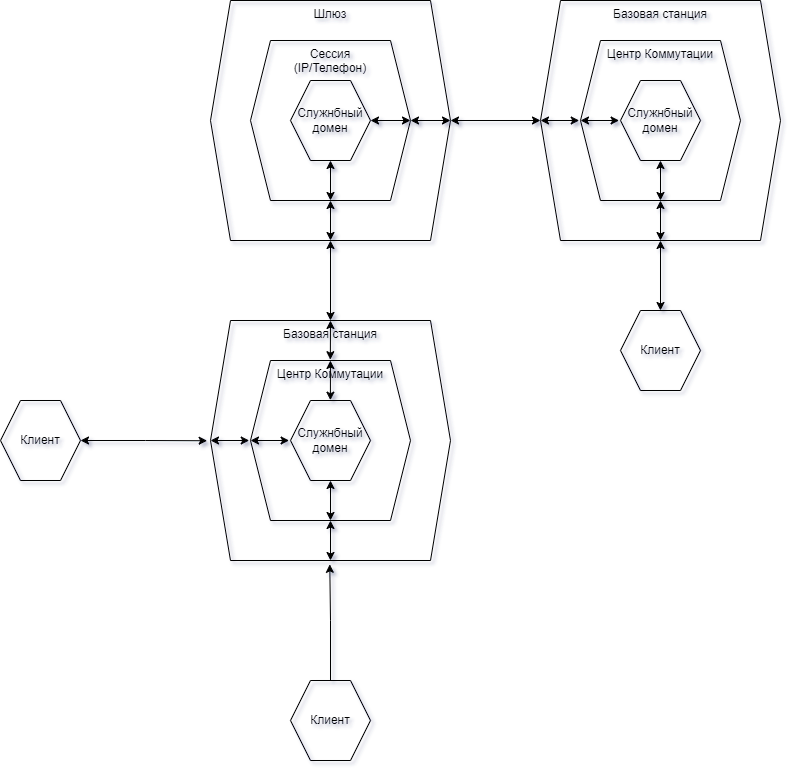


Рисунок 2.1 – Абстрактная схема сети сотовой связи

Данная схема является представлением методологии DDD (Domain Driven Design) или проектирование предметной области, представляет собой методологию разработки программного обеспечения, которая сосредотачивается на понимании и моделировании бизнес-доменов для создания высококачественных и поддерживаемых систем.

Для простоты понимания клиент представлен из одного слоя, хотя в реальности это такой же комплексный домен. А также служебный домен является набором всех служебных функций необходимых для получения сессии – например аутентификация.

Из всего вышеперечисленного можно построить уже немного иную схему для сети LTE.

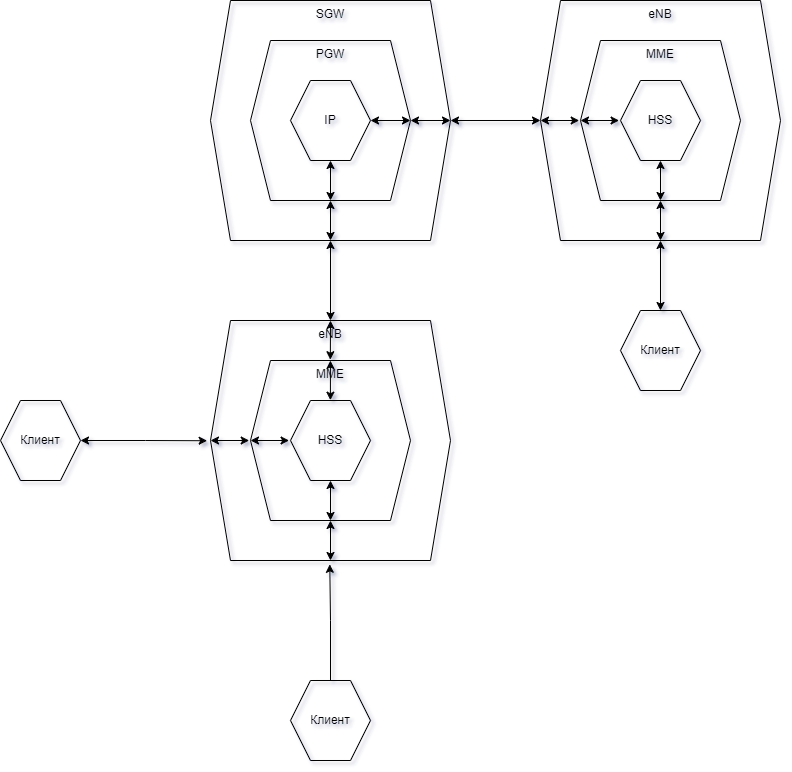


Рисунок 2.2 – DDD LTE сеть

# ПРАКТИЧЕСКАЯ ЧАСТЬ

Реализация эмуляции отправки сообщения будет сделана на языке программирования Python.

Описание всех основных дата-ориентированных классов:

Листинг 3.1 – Класс Client

|  |
| --- |
| **from** dataclasses **import** dataclass, field  **from** src.protocols.protocol **import** Protocol  **from** src.dataclasses.ta **import** TrackingArea  @dataclass(eq=True)  **class** Client:  """Cellular network client"""  client\_id: int  protocols: list[Protocol] = field(compare=False)  available\_ta: TrackingArea = field(compare=False) |

Листинг 3.2 – Класс HSS

|  |
| --- |
| **from** dataclasses **import** dataclass  @dataclass  **class** HSS:  """Home Subscriber Server"""  clients\_ids: list[int] |

Листинг 3.3 – Класс MME

|  |
| --- |
| **from** dataclasses **import** dataclass  **from** src.dataclasses.hss **import** HSS  **from** src.dataclasses.sgw **import** SGW  @dataclass(frozen=True)  **class** MME:  """Mobile Management Entity"""  mme\_id: int  hss: HSS  sgw: list[SGW] |

Листинг 3.4 – Класс PCRF

|  |
| --- |
| **from** dataclasses **import** dataclass  **from** src.dataclasses.pgw **import** PGW  **from** src.dataclasses.pcrf **import** PCRF  **from** src.protocols.protocol **import** Protocol  @dataclass(frozen=True)  **class** SGW:  """Serving GateWay"""  sgw\_id: int  pgw: PGW  pcrf: PCRF    protocols: list[Protocol]  **def** \_\_hash\_\_(self):  **return** hash(self.sgw\_id) |

Листинг 3.5 – Класс TrackingArea

|  |
| --- |
| **From** dataclasses **import** dataclass  **from** src.dataclasses.mme **import** MME  @dataclass(frozen=True)  **class** TrackingArea:  """Tracking Area"""  ta\_id: int  mme: list[MME] |

Листинг 3.6 – Класс Quality

|  |
| --- |
| **from** enum **import** Enum  **class** Quality(Enum):  BAD = 1  NORMAL = 2  GOOD = 3 |

Листинг 3.7 – Класс протокол

|  |
| --- |
| **from** abc **import** ABC, abstractmethod  **class** Protocol(ABC):  """Network protocol"""  @abstractmethod  **def** \_\_init\_\_(self):  """"""    self.name = "name"  **raise** NotImplementedError |

Листинг 3.8 – Класс PDCP

|  |
| --- |
| **from** src.protocols.protocol **import** Protocol  **class** PDCP(Protocol):  """Packet Data Convergence Protocol"""  **def** \_\_init\_\_(self):  """"""    self.name = "PDCP" |

Листинг 3.9 – Класс PLCP

|  |
| --- |
| **from** src.protocols.protocol **import** Protocol  **class** PLCP(Protocol):  """Packet Data Convergence Protocol"""  **def** \_\_init\_\_(self):  """"""    self.name = "PLCP" |

Листинг 3.10 – Класс Recipient

|  |
| --- |
| **from** dataclasses **import** dataclass  @dataclass  **class** Recipient:  """"""    ip: str |

Функционально-ориентированные классы:

Листинг 3.11 – Класс PCRF

|  |
| --- |
| **from** random **import** randint  **from** src.dataclasses.quality **import** Quality  **class** PCRF:  """Policy and Charging Rules Function"""  **def** get\_quality(self) -> Quality:  """"""  **return** Quality(randint(1,3)) |

Листинг 3.12 – Класс Connector

|  |
| --- |
| **from** abc **import** ABC, abstractmethod  **from** src.session.session **import** Session  **class** Connector(ABC):  """Establish Connection"""  @abstractmethod  **def** get\_session() -> Session:  """""" |

Листинг 3.13 – Класс FourthGenerationConnector

|  |
| --- |
| **from** src.connectors.connector **import** Connector  **from** src.session.fourth\_generation\_session **import** FourthGenerationSession  **from** src.dataclasses.sgw **import** SGW  **class** FourthGenerationConnector(Connector):  """"""  **def** get\_session(self, sgw: SGW) -> FourthGenerationSession:  """"""  **return** FourthGenerationSession(sgw=sgw) |

Листинг 3.14 – Класс Session

|  |
| --- |
| **from** abc **import** ABC, abstractmethod  **class** Session(ABC):  """Handling Messages"""  @abstractmethod  **def** send\_message(client\_id: int):  """""" |

Листинг 3.15 – Класс FourthGenerationSession

|  |
| --- |
| **from** src.session.session **import** Session  **from** src.dataclasses.sgw **import** SGW  **from** src.dataclasses.quality **import** Quality  **from** src.dataclasses.recipient **import** Recipient  **from** datetime **import** datetime  **class** FourthGenerationSession(Session):  """"""  **def** \_\_init\_\_(self, sgw: SGW, quality: Quality):  """"""  self.sgw = sgw  self.quality = quality  **print**(f"Session will be using sgw: {self.sgw.sgw\_id}")  **print**(f"Quality will be {self.quality}")      **def** send\_message(self, recipient: Recipient, message: str):  """"""    **print**(f"Message was sent to {recipient.ip} | message: {message}")  **def** \_\_enter\_\_(self):  **print**(f"Session Started | {datetime.utcnow()}")  **return** self    **def** \_\_exit\_\_(self, type, value, traceback):  **print**(f"Session Ended | {datetime.utcnow()}") |

Листинг 3.16 – Класс Controller и его конструктор

|  |
| --- |
| **class** Controller:  """"""  **def** \_\_init\_\_(self, client: Client):  """"""  self.client = client |

Листинг 3.17 – Функция аутентификации и проверки доступности SGW по протоколам

|  |
| --- |
| **def** \_auth(self) -> list[SGW]:  """4G authentication"""  ta = self.client.available\_ta  **for** mme **in** ta.mme:  **if** self.client.client\_id **in** mme.hss.clients\_ids:  sgw = [sgw **for** sgw **in** mme.sgw **if** self.\_\_is\_sgw\_available(sgw)]  **print**(f"After AUTH | found MME id - {mme.mme\_id} | SGW - {[(sgw.sgw\_id) for sgw in sgw]}")  **return** sgw    **def** \_\_is\_sgw\_available(self, sgw: SGW) -> bool:  """"""  **for** protocol **in** self.client.protocols:  **if** protocol.name **in** [protocol.name **for** protocol **in** sgw.protocols]:  **return** True    **return** False |

Листинг 3.18 – Функция получения сессии и выбора SGW из доступных

|  |
| --- |
| **def** \_get\_session(self, sgw: SGW) -> FourthGenerationSession:  """"""  **return** FourthGenerationSession(  sgw=sgw,  quality=sgw.pcrf.get\_quality()  )  **def** \_\_choose\_sgw(self, sgw\_list: list[SGW]) -> SGW:  """"""    **return** sgw\_list[0] **if** sgw\_list **else** None |

Листинг 3.19 – Функция отправки сообщения

|  |
| --- |
| **def** send\_message(self, recipient: Recipient, message: str) -> bool:  """"""  sgw\_list = self.\_auth()  **if** **not** sgw\_list:  **print**("Message was not sent because of your device hasn't needed protocols")  **return** False  sgw = self.\_\_choose\_sgw(sgw\_list=sgw\_list)    **with** self.\_get\_session(sgw) **as** session:  session.send\_message(recipient=recipient, message=message) |

Листинг 3.20 – Основная функция (Entrypoint)

|  |
| --- |
| **from** src.controller **import** Controller  **from** src.storage **import** Clients  **from** src.storage **import** Recipients  **def** main():  """"""  cellular\_device = Controller(client=Clients.client1)  cellular\_device.send\_message(  recipient=Recipients.recipient,  message="Hello!!!"  )  **if** \_\_name\_\_ == "\_\_main\_\_":  main() |

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.1 – Пример двух запусков программы

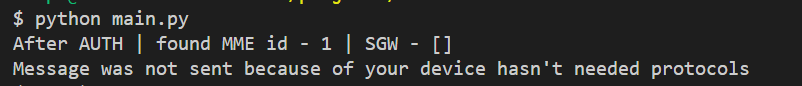


Рисунок 3.2 – Пример неудачного запуска программы, когда доступного SGW по протоколам не оказалось

# ЗАКЛЮЧЕНИЕ

Стандарт 4G (LTE) представляет значительный прогресс в развитии мобильной связи, обеспечивая более высокую скорость передачи данных, улучшенную пропускную способность, низкую задержку и повышенную эффективность сети. Несмотря на свои преимущества, такие как высокая скорость передачи данных, улучшенная пропускная способность и низкая задержка, стандарт 4G также имеет ограничения, включая ограниченное покрытие в отдаленных районах и высокие затраты на инфраструктуру.

Тем не менее, LTE остается ключевым компонентом современных мобильных сетей, играя важную роль в обеспечении доступа к широкому спектру информации и услуг. Технология LTE открывает двери для различных инновационных приложений и сервисов, поддерживая высокую скорость и стабильное подключение.

Обзор архитектуры LTE и применение принципов DDD и SOLID позволяют улучшить понимание и моделирование сетей сотовой связи, обеспечивая более высокое качество и поддерживаемость систем. Принципы DDD и SOLID способствуют созданию гибких и расширяемых систем, где каждый компонент выполняет свою функцию без лишних зависимостей.

Таким образом, LTE остается важным стандартом в мире мобильных коммуникаций, и его развитие отражает стремление к улучшению качества обслуживания и доступности сетей мобильной связи для пользователей по всему миру.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Мэтиз Э. Изучаем Python: программирование игр, визуализация данных, веб – приложения. – 3-е изд. – СПб.: Питер, 2020. – 512с.
2. Eric E. Domain-Driven Design: Tackling Complexity in the Heart of Software. - 1-е издание: Издательство - Addison-Wesley Professional, 2003. – 560 с.
3. Harri H., Antti T. LTE for UMTS: Evolution to LTE-Advanced. - 2-е издание: Издательство – Wiley, 2011. – 576 с.