Изображение выглядит как Графика, графический дизайн, графическая вставка, творческий подход

Автоматически созданное описание

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГАОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  **информационных**  **технологий** | **Кафедра**  **управления и информатики**  **в технических системах** |

**ЛАБОРАТОРНАЯ РАБОТА №1**  
ПО ДИСЦИПЛИНЕ

«ПРИКЛАДНОЕ ПРОГРАММИРОВАНИЕ»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| СТУДЕНТА | 2 | КУРСА | | БАКАЛАВРИАТА | ГРУППЫ | ИДБ-24-14 |
|  | | | *(уровень профессионального образования)* | |  | |

|  |
| --- |
| **ДРОБЯЗКИНА КОНСТАНТИНА АЛЕКСЕЕВИЧА** |
| *(ФИО)* |

ТЕМА РАБОТЫ

|  |
| --- |
| «ООП в Python. Обработка исключительных ситуаций. Форматы XML и JSON» |

|  |  |
| --- | --- |
| Направление: | 09.03.03 Прикладная информатика |

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет сдан «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. | | | |
|  |  |  |  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
|  | | | |
| Преподаватель | Малова Яна Игоревна |  |  |
|  | *(Ф.И.О., должность, степень, звание.)* |  | *(подпись)* |

МОСКВА 2025

Содержание

Оглавление

[Описание работы 3](#_Toc211902281)

[UML-Диаграмма 4](#_Toc211902282)

[Программная реализация 5](#_Toc211902283)

[Листинг программы 5](#_Toc211902284)

[Обработка исключений 35](#_Toc211902285)

[Сериализация и десериализация данных 37](#_Toc211902286)

[Вывод 57](#_Toc211902287)

[Ссылка на удаленный репозиторий 58](#_Toc211902288)

# Описание работы

**Цель работы**: изучить принцип работы ООП в Python, возможности обработки исключительных ситуаций, формирование форматов как XML и JSON.

**Вариант**: предметная область социальной сети.

**Технологии**: python, диаграмма классов UML, формат данных XML и JSON.

**Инструменты**: VSCode / Figma / draw.io

**Задача**:

1. Спроектировать диаграмму классов UML по своему варианту.
2. Реализовать по спроектированной диаграмме классов рабочую программу.
3. Обработать встроенные и собственное исключения. Актуализировать диаграмму классов.
4. Спроектировать структуру для хранения данных в формате XML и JSON.
5. Реализовать считывание из файла (десериализация) и запись в файл (сериализация) всех объектов программы в формат JSON.

# UML-Диаграмма

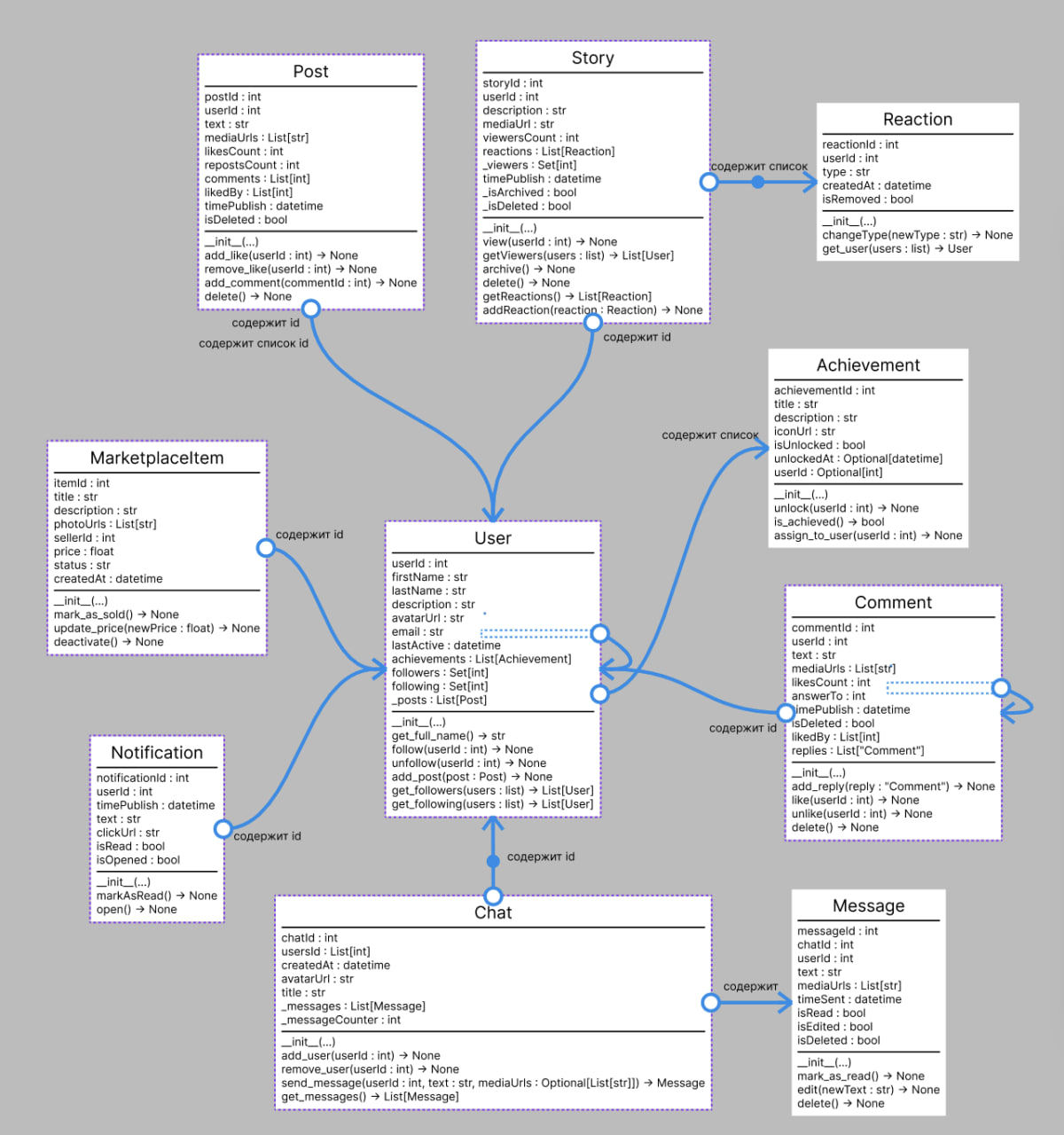


Рисунок 1. UML-Диаграмма (часть 1)

# Программная реализация

## Листинг программы

|  |
| --- |
| Листинг 1. Текст User.py |
| from datetime import datetime  from typing import List, Set  from classes import \*  from custom\_exceptions import \*  class User:      userId: int      firstName: str      lastName: str      description: str      avatarUrl: str      email: str      lastActive: datetime      achievements: List["Achievement"]      followers: Set[int]      following: Set[int]      def \_\_init\_\_(          self,          userId: int,          firstName: str,          lastName: str,          description: str,          avatarUrl: str,          email: str,          lastActive: datetime,          achievements: List["Achievement"] | None = None,          followers: Set[int] | None = None,          following: Set[int] | None = None,      ):          if not isinstance(userId, int):              raise ValueError("userId должен быть целым числом")          if not str(userId).startswith("101"):              raise InvalidUserIdException(userId)          self.userId = userId          if not isinstance(firstName, str) and firstName.strip():              raise ValueError("firstName должен быть непустой строкой.")          self.firstName = firstName          if not isinstance(lastName, str) and lastName.strip():              raise ValueError("lastName должен быть непустой строкой.")          self.lastName = lastName          if not isinstance(description, str):              raise ValueError("description должен быть строкой.")          self.description = description          if not isinstance(avatarUrl, str) or not (              avatarUrl.startswith("http://") or avatarUrl.startswith("https://")          ):              raise ValueError(                  "Некорректный формат avatarUrl (должен начинаться с http:// или https://)."              )          self.avatarUrl = avatarUrl          if (              not isinstance(email, str)              or "@" not in email              or "." not in email.split("@")[-1]          ):              raise InvalidEmailException(email)          self.email = email          if not isinstance(lastActive, datetime):              raise TypeError("lastActive должен быть экземпляром datetime.")          self.lastActive = lastActive          if achievements is not None:              if not isinstance(achievements, list):                  raise TypeError("achievements должен быть списком (list) или None.")              for ach in achievements:                  if not isinstance(ach, Achievement):                      raise TypeError(                          "Все элементы achievements должны быть экземплярами Achievement."                      )          self.achievements = achievements or []          if followers is not None:              if not isinstance(followers, set):                  raise TypeError("followers должен быть множеством (set) или None.")              for f in followers:                  if not isinstance(f, int):                      raise TypeError(                          "Все элементы followers должны быть экземплярами int."                      )          self.followers = followers or set()          if following is not None:              if not isinstance(following, set):                  raise TypeError("following должен быть множеством (set) или None.")              for u in following:                  if not isinstance(u, int):                      raise TypeError(                          "Все элементы following должны быть экземплярами int."                      )          self.following = following or set()      def get\_full\_name(self) -> str:          # Получение полного имени пользователя          return f"{self.firstName} {self.lastName}"      def get\_followers(self) -> List[int]:          # Получение списка ID подписчиков          return list(self.followers)      def get\_following(self) -> List[int]:          # Получение списка ID пользователей, на которых подписан          return list(self.following)      def get\_achievements(self) -> List["Achievement"]:          # Получение списка достижений пользователя          return list(self.achievements)      def update\_profile(          self, description: str | None = None, avatarUrl: str | None = None      ) -> None:          # Обновление описания и аватара пользователя          if description is not None:              self.description = description          if avatarUrl is not None:              self.avatarUrl = avatarUrl          self.lastActive = datetime.now()      def follow(self, user\_id: int) -> None:          # Подписаться на пользователя по ID          if user\_id == self.userId:              return  # Не могу подписаться на самого себя          self.following.add(user\_id)      def unfollow(self, user\_id: int) -> None:          # Отписаться от пользователя по ID          self.following.discard(user\_id)      def add\_follower(self, user\_id: int) -> None:          # Добавить подписчика (используется при импорте или связывании)          if user\_id != self.userId:              self.followers.add(user\_id)      def add\_achievement(self, achievement: "Achievement") -> None:          # Добавить достижение пользователю, если его ещё нет          if achievement not in self.achievements:              self.achievements.append(achievement)      def printing(self) -> None:          print(              f"User(userId={self.userId}, firstName='{self.firstName}', lastName='{self.lastName}', "              f"description='{self.description}', avatarUrl='{self.avatarUrl}', "              f"email='{self.email}', lastActive={self.lastActive}, "              f"achievements={len(self.achievements)}, followers={len(self.followers)}, following={len(self.following)})"          ) |

|  |
| --- |
| Листинг 2. Текст Story.py |
| from datetime import datetime  from typing import List, Set, Optional  from classes import \*  class Story:      storyId: int      userId: int      description: str      mediaUrl: str      viewersCount: int      reactions: List["Reaction"]      viewers: Set[int]      timePublish: datetime      isArchived: bool      isDeleted: bool      def \_\_init\_\_(          self,          storyId: int,          userId: int,          description: str,          mediaUrl: str,          viewersCount: int = 0,          reactions: Optional[List["Reaction"]] = None,          viewers: Optional[Set[int]] = None,          timePublish: Optional[datetime] = None,          isArchived: bool = False,          isDeleted: bool = False,      ):          if not isinstance(storyId, int) or not str(storyId).startswith("110"):              raise TypeError("storyId должен быть int и начинаться с 110.")          self.storyId = storyId          if not isinstance(userId, int):              raise TypeError("userId должен быть int.")          self.userId = userId          if not isinstance(description, str):              raise ValueError("description должен быть строкой.")          self.description = description          if not isinstance(mediaUrl, str) or not (mediaUrl.startswith("http://") or mediaUrl.startswith("https://")):              raise ValueError("Некорректный формат mediaUrl (должен начинаться с http:// или https://).")          self.mediaUrl = mediaUrl          if not isinstance(viewersCount, int) or viewersCount < 0:              raise ValueError("viewersCount должен быть неотрицательным int.")          self.viewersCount = viewersCount          if reactions is not None:              if not isinstance(reactions, list):                  raise TypeError("reactions должен быть списком (list) или None.")              for reaction in reactions:                  if not isinstance(reaction, Reaction):                      raise TypeError("Все элементы reactions должны быть экземплярами Reaction.")          self.reactions = reactions or []          if viewers is not None:              if not isinstance(viewers, set):                  raise TypeError("viewers должен быть множеством (set) или None.")              for viewer in viewers:                  if not isinstance(viewer, int):                      raise TypeError("Все элементы viewers должны быть int.")          self.viewers = viewers or set()          if timePublish is not None and not isinstance(timePublish, datetime):              raise TypeError("timePublish должен быть экземпляром datetime или None.")          self.timePublish = timePublish or datetime.now()          if not isinstance(isArchived, bool):              raise TypeError("isArchived должен быть bool.")          self.isArchived = isArchived          if not isinstance(isDeleted, bool):              raise TypeError("isDeleted должен быть bool.")          self.isDeleted = isDeleted      def view(self, userId: int) -> None:          # Отмечает просмотр истории пользователем          if not self.isDeleted and userId not in self.viewers:              self.viewers.add(userId)              self.viewersCount += 1      def get\_viewers(self, users: List["User"]) -> List["User"]:          # Возвращает список пользователей, просмотревших историю          return [u for u in users if getattr(u, "userId", None) in self.viewers]      def archive(self) -> None:          # Архивирует историю          if not self.isDeleted:              self.isArchived = True      def delete(self) -> None:          # Удаляет историю          self.isDeleted = True          self.description = "[История удалена]"          self.mediaUrl = ""          self.reactions.clear()      def get\_reactions(self) -> List["Reaction"]:          # Возвращает список реакций на историю          return list(self.reactions)      def add\_reaction(self, reaction: "Reaction") -> None:          # Добавляет реакцию к истории          if not self.isDeleted:              self.reactions.append(reaction)      def printing(self) -> None:          print(              f"Story(storyId={self.storyId}, userId={self.userId}, "              f"description='{self.description[:50]}...', mediaUrl='{self.mediaUrl}', "              f"viewersCount={self.viewersCount}, reactions={len(self.reactions)}, "              f"viewers={len(self.viewers)}, timePublish={self.timePublish}, "              f"isArchived={self.isArchived}, isDeleted={self.isDeleted})"          ) |

|  |
| --- |
| Листинг 3. Текст Reaction.py |
| from datetime import datetime  from typing import List, Optional  from classes import \*  class Reaction:      reactionId: int      userId: int      type: str      createdAt: datetime      isRemoved: bool      def \_\_init\_\_(          self,          reactionId: int,          userId: int,          reactionType: str,          createdAt: Optional[datetime] = None,          isRemoved: bool = False,      ):          if not isinstance(reactionId, int) or not str(reactionId).startswith("108"):              raise TypeError("reactionId должен быть int и начинаться с 108.")          self.reactionId = reactionId          if not isinstance(userId, int):              raise TypeError("userId должен быть int.")          self.userId = userId          if not isinstance(reactionType, str) or not reactionType.strip():              raise ValueError("reactionType должен быть непустой строкой.")          self.type = reactionType          if createdAt is not None and not isinstance(createdAt, datetime):              raise TypeError("createdAt должен быть экземпляром datetime или None.")          self.createdAt = createdAt or datetime.now()          if not isinstance(isRemoved, bool):              raise TypeError("isRemoved должен быть bool.")          self.isRemoved = isRemoved      def change\_type(self, newType: str) -> None:          # Изменяем тип реакции (например, с "Сердце" на "Класс")          if not self.isRemoved:              self.type = newType      def remove(self) -> None:          # Удаляем реакцию          self.isRemoved = True      def get\_user(self, users: List["User"]) -> Optional["User"]:          # Получаем объект пользователя по userId          for user in users:              if getattr(user, "userId", None) == self.userId:                  return user          return None      def printing(self) -> None:          print(              f"Reaction(reactionId={self.reactionId}, userId={self.userId}, "              f"type='{self.type}', createdAt={self.createdAt}, isRemoved={self.isRemoved})"          ) |

|  |
| --- |
| Листинг 4. Текст Post.py |
| from datetime import datetime  from typing import List, Optional, Set  class Post:      postId: int      userId: int      text: str      mediaUrls: str      likesCount: int      repostsCount: int      comments: List[int]      timePublish: datetime      isDeleted: bool      likedBy: Set[int]      def \_\_init\_\_(          self,          postId: int,          userId: int,          text: str,          mediaUrls: Optional[List[str]] = None,          likesCount: int = 0,          repostsCount: int = 0,          comments: Optional[List[int]] = None,          timePublish: Optional[datetime] = None,          likedBy: Optional[Set[int]] = None,          isDeleted: bool = False,      ):          if not isinstance(postId, int) or not str(postId).startswith("109"):              raise TypeError("postId должен быть int и начинаться с 109.")          self.postId = postId          if not isinstance(userId, int):              raise TypeError("userId должен быть int.")          self.userId = userId          if not isinstance(text, str):              raise ValueError("text должен быть строкой.")          self.text = text          if not isinstance(mediaUrls, (str, type(None))):              raise TypeError("mediaUrls должен быть строкой или None.")          self.mediaUrls = mediaUrls or None          if not isinstance(likesCount, int) or likesCount < 0:              raise ValueError("likesCount должен быть неотрицательным int.")          self.likesCount = likesCount          if not isinstance(repostsCount, int) or repostsCount < 0:              raise ValueError("repostsCount должен быть неотрицательным int.")          self.repostsCount = repostsCount          if comments is not None and not isinstance(comments, list):              raise TypeError("comments должен быть списком (list) или None.")          self.comments = comments or []          if timePublish is not None and not isinstance(timePublish, datetime):              raise TypeError("timePublish должен быть экземпляром datetime или None.")          self.timePublish = timePublish or datetime.now()          if likedBy is not None and not isinstance(likedBy, set):              raise TypeError("likedBy должен быть множеством (set) или None.")          self.likedBy = likedBy or set()          if not isinstance(isDeleted, bool):              raise TypeError("isDeleted должен быть bool.")          self.isDeleted = isDeleted      def like(self, userId: int) -> None:          # Добавляем лайк, если пост не удалён          if not self.isDeleted and userId not in self.likedBy:              self.likedBy.add(userId)              self.likesCount += 1      def unlike(self, userId: int) -> None:          # Убираем лайк          if userId in self.likedBy:              self.likedBy.remove(userId)              self.likesCount = max(0, self.likesCount - 1)      def add\_comment(self, commentId: int) -> None:          # Добавляем комментарий (по ID)          if not self.isDeleted:              self.comments.append(commentId)      def share(self, userId: int) -> None:          # Увеличиваем счётчик репостов          if not self.isDeleted:              self.repostsCount += 1      def edit(self, text: str, mediaUrls: Optional[List[str]] = None) -> None:          # Редактируем текст и медиа, если пост не удалён          if not self.isDeleted:              self.text = text              if mediaUrls is not None:                  self.mediaUrls = mediaUrls      def delete(self) -> None:          # Удаляем пост          self.text = "[Пост удалён]"          self.mediaUrls.clear()          self.isDeleted = True      def printing(self) -> None:          print(              f"Post(postId={self.postId}, userId={self.userId}, text='{self.text[:50]}...', "              f"mediaUrls={len(self.mediaUrls)}, likesCount={self.likesCount}, "              f"repostsCount={self.repostsCount}, comments={len(self.comments)}, "              f"timePublish={self.timePublish}, isDeleted={self.isDeleted})"          ) |

|  |
| --- |
| Листинг 5. Текст Notification.py |
| from datetime import datetime  from typing import Optional  class Notification:      notificationId: int      userId: int      text: str      clickUrl: str      timePublish: datetime      isRead: bool      isOpened: bool      def \_\_init\_\_(          self,          notificationId: int,          userId: int,          text: str,          clickUrl: str,          timePublish: Optional[datetime] = None,          isRead: bool = False,          isOpened: bool = False,      ):          if not isinstance(notificationId, int) or not str(notificationId).startswith("107"):              raise TypeError("notificationId должен быть int и начинаться с 107.")          self.notificationId = notificationId          if not isinstance(userId, int):              raise TypeError("userId должен быть int.")          self.userId = userId          if not isinstance(text, str):              raise ValueError("text должен быть строкой.")          self.text = text          if not isinstance(clickUrl, str) or not (clickUrl.startswith("http://") or clickUrl.startswith("https://")):              raise ValueError("Некорректный формат clickUrl (должен начинаться с http:// или https://).")          self.clickUrl = clickUrl          if timePublish is not None and not isinstance(timePublish, datetime):              raise TypeError("timePublish должен быть экземпляром datetime или None.")          self.timePublish = timePublish or datetime.now()          if not isinstance(isRead, bool):              raise TypeError("isRead должен быть bool.")          self.isRead = isRead          if not isinstance(isOpened, bool):              raise TypeError("isOpened должен быть bool.")          self.isOpened = isOpened      def mark\_as\_read(self) -> None:          # Помечаем уведомление как прочитанное          self.isRead = True      def open(self) -> None:          # Открываем уведомление и помечаем как прочитанное          self.isOpened = True          self.isRead = True      def printing(self) -> None:          print(              f"Notification(notificationId={self.notificationId}, userId={self.userId}, "              f"text='{self.text[:50]}...', clickUrl='{self.clickUrl}', "              f"timePublish={self.timePublish}, isRead={self.isRead}, isOpened={self.isOpened})"          ) |

|  |
| --- |
| Листинг 6. Текст Message.py |
| from datetime import datetime  from typing import List, Optional  class Message:      messageId: int      chatId: int      userId: int      text: str      mediaUrls: str      timeSent: datetime      isRead: bool      isEdited: bool      isDeleted: bool      def \_\_init\_\_(          self,          messageId: int,          chatId: int,          userId: int,          text: str,          mediaUrls: Optional[List[str]] = None,          timeSent: Optional[datetime] = None,          isRead: bool = False,          isEdited: bool = False,          isDeleted: bool = False,      ):          if not isinstance(messageId, int) or not str(messageId).startswith("106"):              raise TypeError("messageId должен быть int и начинаться с 106.")          self.messageId = messageId          if not isinstance(chatId, int) or not str(chatId).startswith("103"):              raise TypeError("chatId должен быть int и начинаться с 103.")          self.chatId = chatId          if not isinstance(userId, int):              raise TypeError("userId должен быть int.")          self.userId = userId          if not isinstance(text, str):              raise ValueError("text должен быть строкой.")          self.text = text          if not isinstance(mediaUrls, (str, type(None))):              raise TypeError("mediaUrls должен быть строкой или None.")          self.mediaUrls = mediaUrls or None          if timeSent is not None and not isinstance(timeSent, datetime):              raise TypeError("timeSent должен быть экземпляром datetime или None.")          self.timeSent = timeSent or datetime.now()          if not isinstance(isRead, bool):              raise TypeError("isRead должен быть bool.")          self.isRead = isRead          if not isinstance(isEdited, bool):              raise TypeError("isEdited должен быть bool.")          self.isEdited = isEdited          if not isinstance(isDeleted, bool):              raise TypeError("isDeleted должен быть bool.")          self.isDeleted = isDeleted      def edit(self, text: str) -> None:          # Редактируем сообщение, если оно не удалено          if not self.isDeleted:              self.text = text              self.isEdited = True      def delete(self) -> None:          # Помечаем сообщение как удалённое          self.text = "[Сообщение удалено]"          self.mediaUrls.clear()          self.isDeleted = True      def mark\_as\_read(self) -> None:          # Помечаем сообщение как прочитанное          self.isRead = True      def printing(self) -> None:          print(              f"Message(messageId={self.messageId}, chatId={self.chatId}, userId={self.userId}, "              f"text='{self.text[:50]}...', mediaUrls={len(self.mediaUrls)}, "              f"timeSent={self.timeSent}, isRead={self.isRead}, "              f"isEdited={self.isEdited}, isDeleted={self.isDeleted})"          ) |

|  |
| --- |
| Листинг 7. Текст MarketplaceItem.py |
| from datetime import datetime  from typing import List, Optional  from classes import \*  class MarketplaceItem:      itemId: int      title: str      description: str      photoUrls: str      sellerId: int      price: float      status: str  # active или sold      createdAt: datetime      def \_\_init\_\_(          self,          itemId: int,          title: str,          description: str,          photoUrls: Optional[List[str]] = None,          sellerId: int = 0,          price: float = 0.0,          status: str = "active",          createdAt: Optional[datetime] = None,      ):          if not isinstance(itemId, int) or not str(itemId).startswith("105"):              raise TypeError("itemId должен быть int и начинаться с 105.")          self.itemId = itemId          if not isinstance(title, str) or not title.strip():              raise ValueError("title должен быть непустой строкой.")          self.title = title          if not isinstance(description, str):              raise ValueError("description должен быть строкой.")          self.description = description          if not isinstance(photoUrls, (str, type(None))):              raise TypeError("photoUrls должен быть строкой или None.")          self.mediaUrls = photoUrls or None          if not isinstance(sellerId, int):              raise TypeError("sellerId должен быть int.")          self.sellerId = sellerId          if not isinstance(price, (int, float)) or price < 0:              raise ValueError("price должен быть неотрицательным числом.")          self.price = float(price)          if not isinstance(status, str) or status not in ["active", "sold"]:              raise ValueError("status должен быть 'active' или 'sold'.")          self.status = status          if createdAt is not None and not isinstance(createdAt, datetime):              raise TypeError("createdAt должен быть экземпляром datetime или None.")          self.createdAt = createdAt or datetime.now()      def mark\_as\_sold(self) -> None:          # Помечаем товар как проданный          if self.status != "sold":              self.status = "sold"      def add\_photo(self, photoUrl: str) -> None:          # Добавляем новую фотографию товара          self.photoUrls.append(photoUrl)      def update\_price(self, newPrice: float) -> None:          # Обновляем цену, если она положительная          if newPrice > 0:              self.price = newPrice      def get\_seller(self, users: List["User"]) -> Optional["User"]:          # Получаем объект пользователя по sellerId, если он есть в списке          for user in users:              if getattr(user, "userId", None) == self.sellerId:                  return user          return None      def printing(self) -> None:          print(              f"MarketplaceItem(itemId={self.itemId}, title='{self.title}', "              f"description='{self.description[:50]}...', photoUrls={len(self.photoUrls)}, "              f"sellerId={self.sellerId}, price={self.price}, status='{self.status}', "              f"createdAt={self.createdAt})"          ) |

|  |
| --- |
| Листинг 8. Текст Comment.py |
| from datetime import datetime  from typing import List, Optional, Set  from classes import \*  class Comment:      commentId: int      userId: int      text: str      mediaUrls: str      likesCount: int      answerTo: Optional[int]      timePublish: datetime      isDeleted: bool      likedBy: Set[int]      replies: List["Comment"]      def \_\_init\_\_(          self,          commentId: int,          userId: int,          text: str,          likesCount: int,          timePublish: datetime,          mediaUrls: Optional[List[str]] = None,          answerTo: Optional[int] = None,          likedBy: Optional[Set[int]] = None,          isDeleted: bool = False,          replies: Optional[List["Comment"]] = None,      ):          if not isinstance(commentId, int) or not str(commentId).startswith("104"):              raise TypeError("commentId должен быть int и начинаться с 104.")          self.commentId = commentId          if not isinstance(userId, int):              raise TypeError("userId должен быть int.")          self.userId = userId          if not isinstance(text, str):              raise ValueError("text должен быть строкой.")          self.text = text          if not isinstance(likesCount, int) or likesCount < 0:              raise ValueError("likesCount должен быть неотрицательным int.")          self.likesCount = likesCount          if not isinstance(timePublish, datetime):              raise TypeError("timePublish должен быть экземпляром datetime.")          self.timePublish = timePublish          if not isinstance(mediaUrls, (str, type(None))):              raise TypeError("mediaUrls должен быть строкой или None.")          self.mediaUrls = mediaUrls or None          if not isinstance(answerTo, (int, None)):              raise TypeError("answerTo должен быть int или None.")          self.answerTo = answerTo or None          if likedBy is not None and not isinstance(likedBy, set):              raise TypeError("likedBy должен быть множеством (set) или None.")          self.likedBy = likedBy or set()          if not isinstance(isDeleted, bool):              raise TypeError("isDeleted должен быть bool.")          self.isDeleted = isDeleted          if replies is not None and not isinstance(replies, list):              raise TypeError("replies должен быть списком (list) или None.")          self.replies = replies or []      def like(self, userId: int) -> None:          # Лайкаем          if not self.isDeleted and userId not in self.likedBy:              self.likedBy.add(userId)              self.likesCount += 1      def unlike(self, userId: int) -> None:          # Убираем лайк от пользователя          if userId in self.likedBy:              self.likedBy.remove(userId)              self.likesCount = max(0, self.likesCount - 1)      def edit(self, text: str) -> None:          # Изменяем текст комментария, если он не удалён          if not self.isDeleted:              self.text = text      def delete(self) -> None:          # Удаляем комментарий, заменяя текст на placeholder          self.text = "[Комментарий удалён]"          self.mediaUrls.clear()          self.isDeleted = True      def add\_reply(self, reply: "Comment") -> None:          # Добавляем ответ на комментарий          self.replies.append(reply)      def get\_replies(self) -> List["Comment"]:          # Получаем список ответов          return list(self.replies)      def printing(self) -> None:          print(              f"Comment(commentId={self.commentId}, userId={self.userId}, text='{self.text[:50]}...', "              f"mediaUrls={len(self.mediaUrls)}, likesCount={self.likesCount}, "              f"answerTo={self.answerTo}, timePublish={self.timePublish}, "              f"isDeleted={self.isDeleted}, replies={len(self.replies)})"          ) |

|  |
| --- |
| Листинг 9. Текст Chat.py |
| from datetime import datetime  from classes import \*  from typing import List, Optional  class Chat:      chatId: int      usersId: List[int]      createdAt: datetime      avatarUrl: str      title: str      \_messages: List["Message"]      \_messageCounter: int      def \_\_init\_\_(          self,          chatId: int,          usersId: List[int],          avatarUrl: str,          title: str,          createdAt: Optional[datetime] = None,          messages: Optional[List["Message"]] = None,      ):          if not isinstance(chatId, int) or not str(chatId).startswith("103"):              raise TypeError("chatId должен быть int и начинаться с 103.")          self.chatId = chatId          if not isinstance(usersId, list):              raise TypeError("usersId должен быть списком (list).")          for user\_id in usersId:              if not isinstance(user\_id, int):                  raise TypeError("Все элементы usersId должны быть int.")          self.usersId = usersId[:]          if not isinstance(avatarUrl, str) or not (avatarUrl.startswith("http://") or avatarUrl.startswith("https://")):              raise ValueError("Некорректный формат avatarUrl (должен начинаться с http:// или https://).")          self.avatarUrl = avatarUrl          if not isinstance(title, str) or not title.strip():              raise ValueError("title должен быть непустой строкой.")          self.title = title          if createdAt is not None and not isinstance(createdAt, datetime):              raise TypeError("createdAt должен быть экземпляром datetime или None.")          self.createdAt = createdAt or datetime.now()          if messages is not None and not isinstance(messages, list):              raise TypeError("messages должен быть списком (list) или None.")          self.\_messages = messages or []          self.\_messageCounter = len(self.\_messages)      def add\_user(self, userId: int) -> None:          # Добавляем пользователя в чат          if userId not in self.usersId:              self.usersId.append(userId)      def remove\_user(self, userId: int) -> None:          # удаляем пользователя из чата          if userId in self.usersId:              self.usersId.remove(userId)      def send\_message(self, userId: int, text: str, mediaUrls: Optional[List[str]] = None) -> "Message":          # Отправляем сообщение          if userId not in self.usersId:              raise ValueError(f"Ошибка: пользователь {userId} не состоит в чате {self.chatId}")          self.\_messageCounter += 1          msg = Message(              messageId=self.\_messageCounter,              chatId=self.chatId,              userId=userId,              text=text,              mediaUrls=mediaUrls or [],              timeSent=datetime.now(),              isRead=False,              isEdited=False,              isDeleted=False,          )          self.\_messages.append(msg)          return msg      def get\_messages(self) -> List["Message"]:          # Получаем все сообщения          return list(self.\_messages)      def printing(self) -> None:          print(              f"Chat(chatId={self.chatId}, usersId={self.usersId}, "              f"avatarUrl='{self.avatarUrl}', title='{self.title}', "              f"createdAt={self.createdAt}, messages={len(self.\_messages)})"          ) |

|  |
| --- |
| Листинг 10. Текст Achievement.py |
| from datetime import datetime  from classes import \*  from typing import Optional  class Achievement:      achievementId: int      title: str      description: str      iconUrl: str      isUnlocked: bool      unlockedAt: Optional[datetime]      userId: Optional[int]      def \_\_init\_\_(          self,          achievementId: int,          title: str,          description: str,          iconUrl: str,          isUnlocked: bool = False,          unlockedAt: Optional[datetime] = None,          userId: Optional[int] = None,      ):          if not isinstance(achievementId, int) or not str(achievementId).startswith("102"):              raise TypeError("achievementId должен быть int и начинаться с 102.")          self.achievementId = achievementId          if not isinstance(title, str) or not title.strip():              raise ValueError("title должен быть непустой строкой.")          self.title = title          if not isinstance(description, str):              raise ValueError("description должен быть строкой.")          self.description = description          if not isinstance(iconUrl, str) or not (iconUrl.startswith("http://") or iconUrl.startswith("https://")):              raise ValueError("Некорректный формат iconUrl (должен начинаться с http:// или https://).")          self.iconUrl = iconUrl          if not isinstance(isUnlocked, bool):              raise TypeError("isUnlocked должен быть bool.")          self.isUnlocked = isUnlocked          if unlockedAt is not None and not isinstance(unlockedAt, datetime):              raise TypeError("unlockedAt должен быть экземпляром datetime или None.")          self.unlockedAt = unlockedAt          if userId is not None and not isinstance(userId, int):              raise TypeError("userId должен быть int или None.")          self.userId = userId      def unlock(self, userId: int) -> None:          # Помечает достижение как открытое пользователем          if not self.isUnlocked:              self.isUnlocked = True              self.unlockedAt = datetime.now()              self.userId = userId      def is\_achieved(self) -> bool:          # Проверяет, было ли достижение открыто          return self.isUnlocked      def assign\_to\_user(self, userId: int) -> None:          # Привязывает достижение к пользователю (без открытия)          self.userId = userId      def printing(self) -> None:          print(              f"Achievement(achievementId={self.achievementId}, title='{self.title}', "              f"description='{self.description}', iconUrl='{self.iconUrl}', "              f"isUnlocked={self.isUnlocked}, unlockedAt={self.unlockedAt}, userId={self.userId})"          ) |

|  |
| --- |
| Листинг 11. Текст parser.py |
| import json  from datetime import datetime  from typing import List, Optional  from classes import \*  import xml.etree.ElementTree as ET  class Parser:      def parser\_xml(self, path: str = "lab1/data/data.xml") -> None:          tree = ET.parse(path)          root = tree.getroot()          def parse\_section(section\_name: str, item\_name: str):              section = root.find(section\_name)              if section is None:                  return []              result = []              for item in section.findall(item\_name):                  obj = {}                  for child in item:                      tag = child.tag                      text = child.text.strip() if child.text else None                          if tag in ['achievements', 'followers', 'following', 'comments', 'likedBy',                                'mediaUrls', 'reactions', 'viewers', 'usersId', 'replies', 'photoUrls']:                          # Всё для дочерних элементов                          obj[tag] = []                          for sub\_item in child:                              if sub\_item.text and sub\_item.text.strip():                                  obj[tag].append(sub\_item.text.strip())                      else:                          if text == "None" or text == "":                              obj[tag] = None                          else:                              obj[tag] = text                  result.append(obj)              return result          data = {              "users": parse\_section("users", "user"),              "achievements": parse\_section("achievements", "achievement"),              "posts": parse\_section("posts", "post"),              "comments": parse\_section("comments", "comment"),              "reactions": parse\_section("reactions", "reaction"),              "messages": parse\_section("messages", "message"),              "chats": parse\_section("chats", "chat"),              "stories": parse\_section("stories", "story"),              "notifications": parse\_section("notifications", "notification"),              "marketplaceItems": parse\_section("marketplaceItems", "marketplaceItem"),          }          # Преобразование типов данных          for u in data["users"]:              u["userId"] = int(u["userId"])              u["followers"] = set(map(int, u.get("followers", [])))              u["following"] = set(map(int, u.get("following", [])))          for a in data["achievements"]:              a["achievementId"] = int(a["achievementId"])              a["isUnlocked"] = a["isUnlocked"].lower() == "true"              a["userId"] = int(a["userId"]) if a.get("userId") and a["userId"] != "None" else None          for p in data["posts"]:              p["postId"] = int(p["postId"])              p["userId"] = int(p["userId"])              p["likesCount"] = int(p.get("likesCount", 0))              p["repostsCount"] = int(p.get("repostsCount", 0))              p["isDeleted"] = p.get("isDeleted", "false").lower() == "true"              p["likedBy"] = set(map(int, p.get("likedBy", [])))              p["comments"] = [int(x) for x in p.get("comments", [])]              p["mediaUrls"] = p.get("mediaUrls", [])          for c in data["comments"]:              c["commentId"] = int(c["commentId"])              c["userId"] = int(c["userId"])              c["likesCount"] = int(c.get("likesCount", 0))              c["isDeleted"] = c.get("isDeleted", "false").lower() == "true"              c["likedBy"] = set(map(int, c.get("likedBy", [])))              c["mediaUrls"] = c.get("mediaUrls", [])              c["replies"] = c.get("replies", [])              if "answerTo" in c and c["answerTo"] not in (None, ""):                  c["answerTo"] = int(c["answerTo"])              else:                  c["answerTo"] = None          for r in data["reactions"]:              r["reactionId"] = int(r["reactionId"])              r["userId"] = int(r["userId"])              r["isRemoved"] = r.get("isRemoved", "false").lower() == "true"          for m in data["messages"]:              m["messageId"] = int(m["messageId"])              m["chatId"] = int(m["chatId"])              m["userId"] = int(m["userId"])              m["mediaUrls"] = m.get("mediaUrls", [])              for flag in ["isRead", "isEdited", "isDeleted"]:                  m[flag] = m.get(flag, "false").lower() == "true"          for c in data["chats"]:              c["chatId"] = int(c["chatId"])              c["usersId"] = [int(x) for x in c.get("usersId", [])]          for s in data["stories"]:              s["storyId"] = int(s["storyId"])              s["userId"] = int(s["userId"])              s["viewersCount"] = int(s.get("viewersCount", 0))              s["isArchived"] = s.get("isArchived", "false").lower() == "true"              s["isDeleted"] = s.get("isDeleted", "false").lower() == "true"              s["reactions"] = [int(x) for x in s.get("reactions", [])]              s["viewers"] = set(map(int, s.get("viewers", [])))          for n in data["notifications"]:              n["notificationId"] = int(n["notificationId"])              n["userId"] = int(n["userId"])              n["isRead"] = n.get("isRead", "false").lower() == "true"              n["isOpened"] = n.get("isOpened", "false").lower() == "true"          for i in data["marketplaceItems"]:              i["itemId"] = int(i["itemId"])              i["sellerId"] = int(i["sellerId"])              i["price"] = float(i["price"])              i["photoUrls"] = i.get("photoUrls", [])              i["createdAt"] = i["createdAt"]          self.load\_data(data)      def parser\_json(self, path: str = "lab1/data/data.json") -> None:          with open(path, "r", encoding="utf-8") as file:              data = json.load(file)          self.load\_data(data)      def load\_data(self, data):          # Обработка медиа-URL для совместимости с классами          for section in ("posts", "comments", "messages", "marketplaceItems"):              for obj in data.get(section, []):                  key = "photoUrls" if section == "marketplaceItems" else "mediaUrls"                  value = obj.get(key, [])                  if isinstance(value, list):                      obj[key] = ", ".join(str(v) for v in value) if value else ""                  elif value is None:                      obj[key] = ""                  else:                      obj[key] = str(value)          self.users: List[User] = [              User(                  userId=u["userId"],                  firstName=u["firstName"],                  lastName=u["lastName"],                  description=u["description"],                  avatarUrl=u["avatarUrl"],                  email=u["email"],                  lastActive=datetime.fromisoformat(u["lastActive"]),                  achievements=[],                  followers=set(u.get("followers", [])),                  following=set(u.get("following", [])),              )              for u in data["users"]          ]          self.achievements: List[Achievement] = [              Achievement(                  achievementId=a["achievementId"],                  title=a["title"],                  description=a["description"],                  iconUrl=a["iconUrl"],                  isUnlocked=a["isUnlocked"],                  unlockedAt=(                      datetime.fromisoformat(a["unlockedAt"])                      if a.get("unlockedAt")                      else None                  ),                  userId=a.get("userId"),              )              for a in data["achievements"]          ]          self.posts: List[Post] = [              Post(                  postId=p["postId"],                  userId=p["userId"],                  text=p["text"],                  mediaUrls=p.get("mediaUrls", ""),                  likesCount=p.get("likesCount", 0),                  repostsCount=p.get("repostsCount", 0),                  comments=p.get("comments", []),                  likedBy=set(p.get("likedBy", [])),                  timePublish=datetime.fromisoformat(p["timePublish"]),                  isDeleted=p.get("isDeleted", False),              )              for p in data["posts"]          ]          self.comments: List[Comment] = [              Comment(                  commentId=c["commentId"],                  userId=c["userId"],                  text=c["text"],                  mediaUrls=c.get("mediaUrls", ""),                  likesCount=c.get("likesCount", 0),                  answerTo=c.get("answerTo"),                  timePublish=datetime.fromisoformat(c["timePublish"]),                  isDeleted=c.get("isDeleted", False),                  likedBy=set(c.get("likedBy", [])),                  replies=c.get("replies", []),              )              for c in data["comments"]          ]          self.reactions: List[Reaction] = [              Reaction(                  reactionId=r["reactionId"],                  userId=r["userId"],                  reactionType=r["type"],                  createdAt=datetime.fromisoformat(r["createdAt"]),                  isRemoved=r.get("isRemoved", False),              )              for r in data["reactions"]          ]          self.messages: List[Message] = [              Message(                  messageId=m["messageId"],                  chatId=m["chatId"],                  userId=m["userId"],                  text=m["text"],                  mediaUrls=m.get("mediaUrls", ""),                  timeSent=datetime.fromisoformat(m["timeSent"]),                  isRead=m.get("isRead", False),                  isEdited=m.get("isEdited", False),                  isDeleted=m.get("isDeleted", False),              )              for m in data["messages"]          ]          self.chats: List[Chat] = [              Chat(                  chatId=c["chatId"],                  usersId=c["usersId"],                  avatarUrl=c["avatarUrl"],                  title=c["title"],                  createdAt=datetime.fromisoformat(c["createdAt"]),              )              for c in data["chats"]          ]          self.stories: List[Story] = [              Story(                  storyId=s["storyId"],                  userId=s["userId"],                  description=s["description"],                  mediaUrl=s["mediaUrl"],                  viewersCount=s.get("viewersCount", 0),                  reactions=[                      r for r in self.reactions                      if r.reactionId in s.get("reactions", [])                  ],                  viewers=set(s.get("viewers", [])),                  timePublish=datetime.fromisoformat(s["timePublish"]),                  isArchived=s.get("isArchived", False),                  isDeleted=s.get("isDeleted", False),              )              for s in data["stories"]          ]          self.notifications: List[Notification] = [              Notification(                  notificationId=n["notificationId"],                  userId=n["userId"],                  text=n["text"],                  clickUrl=n["clickUrl"],                  timePublish=datetime.fromisoformat(n["timePublish"]),                  isRead=n["isRead"],                  isOpened=n["isOpened"],              )              for n in data["notifications"]          ]          self.marketplaceItems: List[MarketplaceItem] = [              MarketplaceItem(                  itemId=i["itemId"],                  title=i["title"],                  description=i["description"],                  photoUrls=i.get("photoUrls", ""),                  sellerId=i["sellerId"],                  price=i["price"],                  status=i["status"],                  createdAt=datetime.fromisoformat(i["createdAt"]),              )              for i in data["marketplaceItems"]          ]          print(              f"Загружено: "              f"{len(self.users)} пользователей, "              f"{len(self.achievements)} достижений, "              f"{len(self.posts)} постов, "              f"{len(self.comments)} комментариев, "              f"{len(self.reactions)} реакций, "              f"{len(self.messages)} сообщений, "              f"{len(self.chats)} чатов, "              f"{len(self.stories)} историй, "              f"{len(self.notifications)} уведомлений, "              f"{len(self.marketplaceItems)} товаров."          )      def get\_user\_by\_id(self, user\_id: int) -> Optional[User]:          return next((u for u in self.users if u.userId == user\_id), None)      def get\_posts\_by\_user(self, user\_id: int) -> List[Post]:          return [p for p in self.posts if p.userId == user\_id]      def get\_comments\_by\_user(self, user\_id: int) -> List[Comment]:          return [c for c in self.comments if c.userId == user\_id]      def get\_story\_by\_user(self, user\_id: int) -> List[Story]:          return [s for s in self.stories if s.userId == user\_id]      def get\_reactions\_by\_user(self, user\_id: int) -> List[Reaction]:          return [r for r in self.reactions if r.userId == user\_id]      def get\_messages\_by\_user(self, user\_id: int) -> List[Message]:          return [m for m in self.messages if m.userId == user\_id]      def get\_chats\_by\_user(self, user\_id: int) -> List[Chat]:          return [c for c in self.chats if user\_id in c.usersId]      def get\_notifications\_by\_user(self, user\_id: int) -> List[Notification]:          return [n for n in self.notifications if n.userId == user\_id]      def get\_market\_items\_by\_user(self, user\_id: int) -> List[MarketplaceItem]:          return [i for i in self.marketplaceItems if i.sellerId == user\_id]      def get\_achievements\_by\_user(self, user\_id: int) -> List[Achievement]:          return [a for a in self.achievements if a.userId == user\_id]  if \_\_name\_\_ == "\_\_main\_\_":      parser = Parser()      parser.parser\_xml("lab1/data/data.xml")      # Пример вывода      for user in parser.users:          posts = parser.get\_posts\_by\_user(user.userId)          print(f"👤 {user.get\_full\_name()} — постов: {len(posts)}, ")        # parser = Parser()      # parser.parser\_json("data/data.json")      # # Пример вывода      # for user in parser.users:      #     posts = parser.get\_posts\_by\_user(user.userId)      #     print(f"👤 {user.get\_full\_name()} — постов: {len(posts)}, ") |

|  |
| --- |
| Листинг 12. Текст writer.py |
| import json  from datetime import datetime  from typing import List, Optional  from classes import \*  from parser import Parser  import xml.etree.ElementTree as ET  class Writer:      def \_\_init\_\_(self, parser: Parser):          self.parser = parser      def to\_json(self, path: str = "lab1/data/written\_data.json") -> None:          def serialize(obj):              if isinstance(obj, datetime):                  return obj.isoformat()              if isinstance(obj, set):                  return list(obj)              if hasattr(obj, "\_\_dict\_\_"):                  return obj.\_\_dict\_\_              return str(obj)          data = {              "users": self.parser.users,              "achievements": self.parser.achievements,              "posts": self.parser.posts,              "comments": self.parser.comments,              "reactions": self.parser.reactions,              "messages": self.parser.messages,              "chats": self.parser.chats,              "stories": self.parser.stories,              "notifications": self.parser.notifications,              "marketplaceItems": self.parser.marketplaceItems,          }            with open(path, "w", encoding="utf-8") as f:              json.dump(data, f, ensure\_ascii=False, indent=4, default=serialize)          print(f"Данные успешно записаны в {path}")        def to\_xml(self, path: str = "lab1/data/written\_data.xml") -> None:          # Сохраняет данные с правильной структурой массивов          def serialize\_value(value):              if isinstance(value, datetime):                  return value.isoformat()              if isinstance(value, (list, set, tuple)):                  return ", ".join(map(str, value))              return str(value)          def build\_element(parent, key, value):              # Добавляет элементы XML              if isinstance(value, dict):                  elem = ET.SubElement(parent, key)                  for k, v in value.items():                      build\_element(elem, k, v)              elif isinstance(value, list):                  list\_elem = ET.SubElement(parent, key)                  for item in value:                      item\_tag = key[:-1] if key.endswith("s") else "item"                      build\_element(list\_elem, item\_tag, item)              elif hasattr(value, "\_\_dict\_\_"):                  obj\_elem = ET.SubElement(parent, key)                  for k, v in value.\_\_dict\_\_.items():                      build\_element(obj\_elem, k, v)              else:                  elem = ET.SubElement(parent, key)                  elem.text = serialize\_value(value)          def build\_object\_element(parent, obj, item\_tag):              # Создает элемент объекта с правильной структурой массивов              obj\_elem = ET.SubElement(parent, item\_tag)                for attr\_name, attr\_value in obj.\_\_dict\_\_.items():                  if attr\_name in ['achievements', 'followers', 'following', 'comments', 'likedBy',                                 'mediaUrls', 'reactions', 'viewers', 'usersId', 'replies', 'photoUrls']:                      # Это массив, создаем контейнер                      array\_elem = ET.SubElement(obj\_elem, attr\_name)                      if isinstance(attr\_value, (list, set, tuple)) and attr\_value:                          for item in attr\_value:                              item\_elem = ET.SubElement(array\_elem, attr\_name[:-1] if attr\_name.endswith("s") else "item")                              # Если это объект Reaction, берем его ID                              if hasattr(item, 'reactionId'):                                  item\_elem.text = str(item.reactionId)                              else:                                  item\_elem.text = str(item)                      elif isinstance(attr\_value, str) and attr\_value:                          # Если это строка с разделителями (например, mediaUrls)                          items = [item.strip() for item in attr\_value.split(",") if item.strip()]                          for item in items:                              item\_elem = ET.SubElement(array\_elem, "url" if attr\_name == "mediaUrls" or attr\_name == "photoUrls" else "item")                              item\_elem.text = item                  else:                      # Обычное поле                      elem = ET.SubElement(obj\_elem, attr\_name)                      elem.text = serialize\_value(attr\_value)                return obj\_elem          # Корневой элемент XML          root = ET.Element("data")          # Секции данных          sections = {              "users": (self.parser.users, "user"),              "achievements": (self.parser.achievements, "achievement"),              "posts": (self.parser.posts, "post"),              "comments": (self.parser.comments, "comment"),              "reactions": (self.parser.reactions, "reaction"),              "messages": (self.parser.messages, "message"),              "chats": (self.parser.chats, "chat"),              "stories": (self.parser.stories, "story"),              "notifications": (self.parser.notifications, "notification"),              "marketplaceItems": (self.parser.marketplaceItems, "marketplaceItem"),          }          # Заполняем XML          for section\_name, (items, item\_tag) in sections.items():              section\_el = ET.SubElement(root, section\_name)              for item in items:                  build\_object\_element(section\_el, item, item\_tag)          # Создаем и сохраняем XML-файл          tree = ET.ElementTree(root)          tree.write(path, encoding="utf-8", xml\_declaration=True)          print(f"Данные успешно записаны в {path}")    if \_\_name\_\_ == "\_\_main\_\_":      parser = Parser()        print("=========================Парсинг исходного XML======================\n")      parser.parser\_xml("lab1/data/data.xml")        myWriter = Writer(parser)        print("Запись в JSON...")      myWriter.to\_json("lab1/data/written\_data.json")        print("Запись в XML...")      myWriter.to\_xml("lab1/data/written\_data.xml")        print("\n-----------------------Тестирование парсинга записанных файлов------------------------\n")        parser2 = Parser()      parser2.parser\_json("lab1/data/written\_data.json")      print("с всё JSON норм")        parser3 = Parser()      parser3.parser\_xml("lab1/data/written\_data.xml")      print("с XML всё норм") |

## Обработка исключений

|  |
| --- |
| custom\_exceptions.py |
| class InvalidUserIdException(Exception):        def \_\_init\_\_(self, user\_id, message="Некорректный ID пользователя"):          self.user\_id = user\_id          self.message = f"{message}: {user\_id}. ID должен начинаться с '101' и быть числом."          super().\_\_init\_\_(self.message)  class InvalidEmailException(Exception):        def \_\_init\_\_(self, email, message="Некорректный email адрес"):          self.email = email          self.message = f"{message}: {email}. Email должен содержать '@' и домен с точкой."          super().\_\_init\_\_(self.message) |

|  |
| --- |
| exeptions.py |
| from datetime import datetime  from classes import \*  def test\_exceptions():      print("Тест создания класса с невалидными данными\n")      try:          user1 = User(              userId=101001,              firstName="Иван",              lastName="Петров",              description="Тест",              avatarUrl="https://example.com/avatar.jpg",              email="Привет всем! это мой влог",  # Невалидный email              lastActive=datetime.now(),          )          print(f"   ✅ Пользователь создан: {user1.get\_full\_name()}")      except Exception as e:          print(f"   ❌ Ошибка: {type(e).\_\_name\_\_}: {e}")      print("======================================\n")      try:          user2 = User(              userId=104002,  # Невалидный ID              firstName="Мария",              lastName="Иванова",              description="Тест",              avatarUrl="https://example.com/avatar2.jpg",              email="123@valid.com",              lastActive=datetime.now(),              followers={101404},          )          print(f"   ✅ Пользователь создан: {user2.get\_full\_name()}")      except Exception as e:          print(f"   ❌ Ошибка: {type(e).\_\_name\_\_}: {e}")      print("======================================\n")      try:          user2 = User(              userId="Это мой id",  # Невалидный ID              firstName="Мария",              lastName="Иванова",              description="Тест",              avatarUrl="https://example.com/avatar2.jpg",              email="123@valid.com",              lastActive=datetime.now(),              followers={101404},          )          print(f"   ✅ Пользователь создан: {user2.get\_full\_name()}")      except Exception as e:          print(f"   ❌ Ошибка: {type(e).\_\_name\_\_}: {e}")      print("======================================\n")      try:          user2 = User(              userId=101002,              firstName="Мария",              lastName="Иванова",              description="Тест",              avatarUrl="https://example.com/avatar2.jpg",              email="123@valid.com",              lastActive=datetime.now(),              followers={101404},          )          print(f"   ✅ Пользователь создан: {user2.get\_full\_name()}")      except Exception as e:          print(f"   ❌ Ошибка: {type(e).\_\_name\_\_}: {e}")  test\_exceptions() |

|  |
| --- |
| Терминал |
| С:/../lab1/exeptions.py  Тест создания класса с невалидными данными  ❌ Ошибка: ValueError: Некорректный формат email.  ======================================  ❌ Ошибка: TypeError: userId должен быть int и начинаться с 101. |

# Сериализация и десериализация данных

|  |
| --- |
| data.json пользователи |
| "users": [      {        "userId": 101001,        "firstName": "Иван",        "lastName": "Дроздов",        "description": "Мне 19, люблю играть в компьютерные игры",        "avatarUrl": "https://avatars.mds.yandex.net/i?id=ed2a6faf31cdeeca89cffcba1d4fb7a4\_l-10696367-images-thumbs&n=13",        "email": "example1@gmail.com",        "lastActive": "2025-10-09T23:03:00",        "achievements": [          102000,          102001,          102002        ],        "followers": [          101004        ],        "following": [          101002        ]      },      {        "userId": 101002,        "firstName": "Петр",        "lastName": "Сидоров",        "description": "Я преподаватель математики в школе",        "avatarUrl": "https://i.pinimg.com/originals/ef/2a/af/ef2aaf4a43cc6293fc49f43716800a98.jpg",        "email": "example2@gmail.com",        "lastActive": "2025-10-10T23:59:00",        "achievements": [          102000,          102001        ],        "followers": [          101001        ],        "following": [          101004        ]      },      {        "userId": 101003,        "firstName": "Василий",        "lastName": "Медведев",        "description": "Сочиняю свою музыку в стиле репа",        "avatarUrl": "https://avatars.mds.yandex.net/i?id=856af239789ab3f5f7962897c9a69647\_l-12422990-images-thumbs&n=13",        "email": "example3@gmail.com",        "lastActive": "2025-10-11T13:04:00",        "achievements": [          102000,          102003        ],        "followers": [],        "following": [          101001        ]      },      {        "userId": 101004,        "firstName": "Федор",        "lastName": "Васильев",        "description": "Как поймать тигра в клетку? Никак. Тигры бывают только в полоску",        "avatarUrl": "https://i.pinimg.com/originals/19/9b/9c/199b9cf3b7d032b39fc5c7e920fd625c.jpg",        "email": "example4@gmail.com",        "lastActive": "2025-10-11T13:04:00",        "achievements": [          102000,          102001        ],        "followers": [          101001,          101002        ],        "following": []      }    ], |

|  |
| --- |
| data.json достижения |
| "achievements": [      {        "achievementId": 102000,        "title": "Создатель",        "description": "Создать аккаунт в социальной сети cheburashki.net",        "iconUrl": "https://avatars.mds.yandex.net/i?id=e907352b44c15222bec671beb353abaa\_l-5878053-images-thumbs&n=13",        "isUnlocked": true,        "unlockedAt": "2025-10-09T23:03:00",        "userId": 101001      },      {        "achievementId": 102001,        "title": "Общительный",        "description": "Вступить в беседу",        "iconUrl": "https://avatars.mds.yandex.net/i?id=e66c556ac8eeb056e00d294eb5cb8b55\_l-5258850-images-thumbs&n=13",        "isUnlocked": false,        "unlockedAt": null,        "userId": null      },      {        "achievementId": 102002,        "title": "Популярный",        "description": "Иметь на посту 2 и более лайков и хотя бы 1 комментарий",        "iconUrl": "https://avatars.mds.yandex.net/i?id=6092f1254884c8e45e81b513b3a0a1f1\_l-5213641-images-thumbs&n=13",        "isUnlocked": false,        "unlockedAt": null,        "userId": null      },      {        "achievementId": 102003,        "title": "Создатель контента",        "description": "Опубликовать историю",        "iconUrl": "https://avatars.mds.yandex.net/i?id=08a147d36d90289b0fa99128a397906a4f4b5af3-9264516-images-thumbs&n=13",        "isUnlocked": true,        "unlockedAt": "2025-10-11T11:00:00",        "userId": 101003      }    ], |

|  |
| --- |
| data.json посты |
| "posts": [      {        "postId": 109001,        "userId": 101001,        "text": "Всем привет! Это мой первый пост.",        "mediaUrls": [],        "likesCount": 2,        "repostsCount": 1,        "comments": [          104001        ],        "likedBy": [          101004,          101003        ],        "timePublish": "2025-10-11T08:31:00",        "isDeleted": false      },      {        "postId": 109002,        "userId": 101002,        "text": "Тигр — полосатый!",        "mediaUrls": [],        "likesCount": 0,        "repostsCount": 0,        "comments": [          104002        ],        "likedBy": [],        "timePublish": "2025-10-11T08:33:00",        "isDeleted": false      },      {        "postId": 109003,        "userId": 101003,        "text": "Обожаю котиков",        "mediaUrls": [          "https://img.freepik.com/premium-photo/cuties-cat-ever-looking-you-wallpaper\_974726-637.jpg"        ],        "likesCount": 2,        "repostsCount": 0,        "comments": [          104003        ],        "likedBy": [          101001,          101004        ],        "timePublish": "2025-10-11T07:50:00",        "isDeleted": false      },      {        "postId": 109004,        "userId": 101004,        "text": "Кто в Клеш?",        "mediaUrls": [],        "likesCount": 1,        "repostsCount": 1,        "comments": [],        "likedBy": [          101001        ],        "timePublish": "2025-10-11T09:00:00",        "isDeleted": false      }    ], |

|  |
| --- |
| data.json комментарии |
| "comments": [      {        "commentId": 104001,        "userId": 101004,        "text": "Бомба супер класс!",        "mediaUrls": [],        "likesCount": 0,        "answerTo": 109001,        "timePublish": "2025-10-11T08:34:00",        "isDeleted": false,        "likedBy": [],        "replies": []      },      {        "commentId": 104002,        "userId": 101001,        "text": "Тигр",        "mediaUrls": [],        "likesCount": 1,        "answerTo": 109002,        "timePublish": "2025-10-11T08:32:00",        "isDeleted": false,        "likedBy": [          101001        ],        "replies": []      },      {        "commentId": 104003,        "userId": 101002,        "text": "Обожаю котиков",        "mediaUrls": [],        "likesCount": 2,        "answerTo": 109003,        "timePublish": "2025-10-11T07:50:00",        "isDeleted": false,        "likedBy": [          101001,          101004        ],        "replies": []      }    ], |

|  |
| --- |
| data.json реакции |
| "reactions": [      {        "reactionId": 108001,        "userId": 101001,        "type": "Сердце",        "createdAt": "2025-10-10T10:00:00",        "isRemoved": false      },      {        "reactionId": 108002,        "userId": 101003,        "type": "Класс",        "createdAt": "2025-10-11T12:00:00",        "isRemoved": false      },      {        "reactionId": 108003,        "userId": 101004,        "type": "Класс",        "createdAt": "2025-10-12T09:15:00",        "isRemoved": false      }    ], |

|  |
| --- |
| data.json сообщения |
| "messages": [      {        "messageId": 106001,        "chatId": 103001,        "userId": 101001,        "text": "ВСем привет",        "mediaUrls": [],        "timeSent": "2025-10-11T08:30:00",        "isRead": false,        "isEdited": false,        "isDeleted": false      },      {        "messageId": 106002,        "chatId": 103001,        "userId": 101001,        "text": "Как дела?",        "mediaUrls": [],        "timeSent": "2025-10-11T08:45:00",        "isRead": false,        "isEdited": false,        "isDeleted": false      },      {        "messageId": 106003,        "chatId": 103001,        "userId": 101002,        "text": "Всё хорошо",        "mediaUrls": [],        "timeSent": "2025-10-11T09:10:00",        "isRead": false,        "isEdited": false,        "isDeleted": false      },      {        "messageId": 106004,        "chatId": 103002,        "userId": 101003,        "text": "А как лабу делать?",        "mediaUrls": [],        "timeSent": "2025-10-11T13:00:00",        "isRead": false,        "isEdited": false,        "isDeleted": false      },      {        "messageId": 106005,        "chatId": 103002,        "userId": 101004,        "text": "Чд?",        "mediaUrls": [],        "timeSent": "2025-10-11T14:55:00",        "isRead": false,        "isEdited": false,        "isDeleted": false      }    ], |

|  |
| --- |
| data.json чаты |
| "chats": [      {        "chatId": 103001,        "usersId": [          101001,          101002,          101003,          101004        ],        "createdAt": "2025-10-01T12:00:00",        "avatarUrl": "https://static0.gamerantimages.com/wordpress/wp-content/uploads/2024/07/clash-royale-cards.jpg?q=49&fit=crop&w=825&dpr=2",        "title": "Клеш Рояль"      },      {        "chatId": 103002,        "usersId": [          101002,          101003        ],        "createdAt": "2025-10-05T18:30:00",        "avatarUrl": "https://i.pinimg.com/736x/80/64/4a/80644aeb5df244c0b4b722cfa1e0b47b.jpg",        "title": "Bestики"      },      {        "chatId": 103003,        "usersId": [          101001,          101004        ],        "createdAt": "2025-10-07T09:00:00",        "avatarUrl": "https://avatars.mds.yandex.net/i?id=0168e824d01d542d52d9472033815d61\_sr-5859172-images-thumbs&n=13",        "title": "Кабанчики"      }    ], |

|  |
| --- |
| data.json истории |
| "stories": [      {        "storyId": 110001,        "userId": 101003,        "description": "Новая демка",        "mediaUrl": "https://example.com/story1.mp4",        "timePublish": "2025-10-11T11:00:00",        "viewersCount": 1,        "reactions": [          108001        ],        "viewers": [          101001        ],        "isArchived": false,        "isDeleted": false      },      {        "storyId": 110002,        "userId": 101001,        "description": "Катка в CR",        "mediaUrl": "https://example.com/story2.mp4",        "timePublish": "2025-10-11T12:10:00",        "viewersCount": 0,        "reactions": [],        "viewers": [],        "isArchived": false,        "isDeleted": false      }    ], |

|  |
| --- |
| data.json уведомления |
| "notifications": [      {        "notificationId": 107001,        "userId": 101001,        "text": "ВСем привет",        "clickUrl": "https://cheburashki.net/chats/103001",        "timePublish": "2025-10-11T08:30:00",        "isRead": false,        "isOpened": false      },      {        "notificationId": 107002,        "userId": 101003,        "text": "А как лабу делать?",        "clickUrl": "https://cheburashki.net/chats/103002",        "timePublish": "2025-10-11T13:00:00",        "isRead": false,        "isOpened": false      }    ], |

|  |
| --- |
| data.json товар на маркетплейс |
| "marketplaceItems": [      {        "itemId": 105001,        "title": "Лабуба ориг",        "description": "Оригинальная Лабуба прямо из Китая, досталась по наследству",        "photoUrls": [          "https://avatars.dzeninfra.ru/get-zen\_doc/271828/pub\_68e3d9bc98ee0c2fb7eee9f0\_68e3d9cccbdb0445cf0b2cbb/scale\_1200"        ],        "sellerId": 101001,        "price": 40000,        "status": "active",        "createdAt": "2025-03-10T00:00:00"      },      {        "itemId": 105002,        "title": "Утюг",        "description": "Старинный утюг времён варягов и греков",        "photoUrls": [          "https://avatars.mds.yandex.net/i?id=ddc8952e84a2e1f0cdf1780484390556\_l-3524219-images-thumbs&n=13"        ],        "sellerId": 101001,        "price": 300,        "status": "active",        "createdAt": "2023-09-30T00:00:00"      },      {        "itemId": 105003,        "title": "Велосипед",        "description": "Алюминиевый велосипед Stels",        "photoUrls": [          "https://avatars.mds.yandex.net/get-mpic/5335973/2a00000191e501bdb77f5948b454b8bd9950/9hq"        ],        "sellerId": 101004,        "price": 10000,        "status": "sold",        "createdAt": "2024-06-15T00:00:00"      }    ] |

|  |
| --- |
| data.xml пользователи |
| <users>          <user>              <userId>101001</userId>              <firstName>Иван</firstName>              <lastName>Дроздов</lastName>              <description>Мне 19, люблю играть в компьютерные игры</description>              <avatarUrl>                  https://avatars.mds.yandex.net/i?id=ed2a6faf31cdeeca89cffcba1d4fb7a4\_l-10696367-images-thumbs&amp;n=13</avatarUrl>              <email>example1@gmail.com</email>              <lastActive>2025-10-09T23:03:00</lastActive>              <achievements>                  <achievement>102000</achievement>                  <achievement>102001</achievement>                  <achievement>102002</achievement>              </achievements>              <followers>                  <follower>101004</follower>              </followers>              <following>                  <follow>101002</follow>              </following>          </user>          <user>              <userId>101002</userId>              <firstName>Петр</firstName>              <lastName>Сидоров</lastName>              <description>Я преподаватель математики в школе</description>              <avatarUrl>https://i.pinimg.com/originals/ef/2a/af/ef2aaf4a43cc6293fc49f43716800a98.jpg</avatarUrl>              <email>example2@gmail.com</email>              <lastActive>2025-10-10T23:59:00</lastActive>              <achievements>                  <achievement>102000</achievement>                  <achievement>102001</achievement>              </achievements>              <followers>                  <follower>101001</follower>              </followers>              <following>                  <follow>101004</follow>              </following>          </user>          <user>              <userId>101003</userId>              <firstName>Василий</firstName>              <lastName>Медведев</lastName>              <description>Сочиняю свою музыку в стиле репа</description>              <avatarUrl>                  https://avatars.mds.yandex.net/i?id=856af239789ab3f5f7962897c9a69647\_l-12422990-images-thumbs&amp;n=13</avatarUrl>              <email>example3@gmail.com</email>              <lastActive>2025-10-11T13:04:00</lastActive>              <achievements>                  <achievement>102000</achievement>                  <achievement>102003</achievement>              </achievements>              <followers>              </followers>              <following>                  <follow>101001</follow>              </following>          </user>          <user>              <userId>101004</userId>              <firstName>Федор</firstName>              <lastName>Васильев</lastName>              <description>Как поймать тигра в клетку? Никак. Тигры бывают только в полоску</description>              <avatarUrl>https://i.pinimg.com/originals/19/9b/9c/199b9cf3b7d032b39fc5c7e920fd625c.jpg</avatarUrl>              <email>example4@gmail.com</email>              <lastActive>2025-10-11T13:04:00</lastActive>              <achievements>                  <achievement>102000</achievement>                  <achievement>102001</achievement>              </achievements>              <followers>                  <follower>101001</follower>                  <follower>101002</follower>              </followers>              <following>              </following>          </user>      </users> |

|  |
| --- |
| data.xml достижения |
| <achievements>          <achievement>              <achievementId>102000</achievementId>              <title>Создатель</title>              <description>Создать аккаунт в социальной сети cheburashki.net</description>              <iconUrl>                  https://avatars.mds.yandex.net/i?id=e907352b44c15222bec671beb353abaa\_l-5878053-images-thumbs&amp;n=13</iconUrl>              <isUnlocked>true</isUnlocked>              <unlockedAt>2025-10-09T23:03:00</unlockedAt>              <userId>101001</userId>          </achievement>          <achievement>              <achievementId>102001</achievementId>              <title>Общительный</title>              <description>Вступить в беседу</description>              <iconUrl>                  https://avatars.mds.yandex.net/i?id=e66c556ac8eeb056e00d294eb5cb8b55\_l-5258850-images-thumbs&amp;n=13</iconUrl>              <isUnlocked>false</isUnlocked>              <unlockedAt />              <userId />          </achievement>          <achievement>              <achievementId>102002</achievementId>              <title>Популярный</title>              <description>Иметь на посту 2 и более лайков и хотя бы 1 комментарий</description>              <iconUrl>                  https://avatars.mds.yandex.net/i?id=6092f1254884c8e45e81b513b3a0a1f1\_l-5213641-images-thumbs&amp;n=13</iconUrl>              <isUnlocked>false</isUnlocked>              <unlockedAt />              <userId />          </achievement>          <achievement>              <achievementId>102003</achievementId>              <title>Создатель контента</title>              <description>Опубликовать историю</description>              <iconUrl>                  https://avatars.mds.yandex.net/i?id=08a147d36d90289b0fa99128a397906a4f4b5af3-9264516-images-thumbs&amp;n=13</iconUrl>              <isUnlocked>true</isUnlocked>              <unlockedAt>2025-10-11T11:00:00</unlockedAt>              <userId>101003</userId>          </achievement>      </achievements> |

|  |
| --- |
| data.xml посты |
| <posts>          <post>              <postId>109001</postId>              <userId>101001</userId>              <text>Всем привет! Это мой первый пост.</text>              <mediaUrls>              </mediaUrls>              <likesCount>2</likesCount>              <repostsCount>1</repostsCount>              <comments>                  <comment>104001</comment>              </comments>              <likedBy>                  <user>101004</user>                  <user>101003</user>              </likedBy>              <timePublish>2025-10-11T08:31:00</timePublish>              <isDeleted>false</isDeleted>          </post>          <post>              <postId>109002</postId>              <userId>101002</userId>              <text>Тигр — полосатый!</text>              <mediaUrls>              </mediaUrls>              <likesCount>0</likesCount>              <repostsCount>0</repostsCount>              <comments>                  <comment>104002</comment>              </comments>              <likedBy>              </likedBy>              <timePublish>2025-10-11T08:33:00</timePublish>              <isDeleted>false</isDeleted>          </post>          <post>              <postId>109003</postId>              <userId>101003</userId>              <text>Обожаю котиков</text>              <mediaUrls>                  <url>https://img.freepik.com/premium-photo/cuties-cat-ever-looking-you-wallpaper\_974726-637.jpg</url>              </mediaUrls>              <likesCount>2</likesCount>              <repostsCount>0</repostsCount>              <comments>                  <comment>104003</comment>              </comments>              <likedBy>                  <user>101001</user>                  <user>101004</user>              </likedBy>              <timePublish>2025-10-11T07:50:00</timePublish>              <isDeleted>false</isDeleted>          </post>          <post>              <postId>109004</postId>              <userId>101004</userId>              <text>Кто в Клеш?</text>              <mediaUrls>              </mediaUrls>              <likesCount>1</likesCount>              <repostsCount>1</repostsCount>              <comments>              </comments>              <likedBy>                  <user>101001</user>              </likedBy>              <timePublish>2025-10-11T09:00:00</timePublish>              <isDeleted>false</isDeleted>          </post>      </posts> |

|  |
| --- |
| data.xml комментарии |
| <comments>          <comment>              <commentId>104001</commentId>              <userId>101004</userId>              <text>Бомба супер класс!</text>              <mediaUrls>              </mediaUrls>              <likesCount>0</likesCount>              <answerTo>109001</answerTo>              <timePublish>2025-10-11T08:34:00</timePublish>              <isDeleted>false</isDeleted>              <likedBy>              </likedBy>              <replies>              </replies>          </comment>          <comment>              <commentId>104002</commentId>              <userId>101001</userId>              <text>Тигр</text>              <mediaUrls>              </mediaUrls>              <likesCount>1</likesCount>              <answerTo>109002</answerTo>              <timePublish>2025-10-11T08:32:00</timePublish>              <isDeleted>false</isDeleted>              <likedBy>                  <user>101001</user>              </likedBy>              <replies>              </replies>          </comment>          <comment>              <commentId>104003</commentId>              <userId>101002</userId>              <text>Обожаю котиков</text>              <mediaUrls>              </mediaUrls>              <likesCount>2</likesCount>              <answerTo>109003</answerTo>              <timePublish>2025-10-11T07:50:00</timePublish>              <isDeleted>false</isDeleted>              <likedBy>                  <user>101001</user>                  <user>101004</user>              </likedBy>              <replies>              </replies>          </comment>      </comments> |

|  |
| --- |
| data.xml реакции |
| <reactions>          <reaction>              <reactionId>108001</reactionId>              <userId>101001</userId>              <type>Сердце</type>              <createdAt>2025-10-10T10:00:00</createdAt>              <isRemoved>false</isRemoved>          </reaction>          <reaction>              <reactionId>108002</reactionId>              <userId>101003</userId>              <type>Класс</type>              <createdAt>2025-10-11T12:00:00</createdAt>              <isRemoved>false</isRemoved>          </reaction>          <reaction>              <reactionId>108003</reactionId>              <userId>101004</userId>              <type>Класс</type>              <createdAt>2025-10-12T09:15:00</createdAt>              <isRemoved>false</isRemoved>          </reaction>      </reactions> |

|  |
| --- |
| data.xml сообщения |
| <messages>          <message>              <messageId>106001</messageId>              <chatId>103001</chatId>              <userId>101001</userId>              <text>ВСем привет</text>              <mediaUrls>              </mediaUrls>              <timeSent>2025-10-11T08:30:00</timeSent>              <isRead>false</isRead>              <isEdited>false</isEdited>              <isDeleted>false</isDeleted>          </message>          <message>              <messageId>106002</messageId>              <chatId>103001</chatId>              <userId>101001</userId>              <text>Как дела?</text>              <mediaUrls>              </mediaUrls>              <timeSent>2025-10-11T08:45:00</timeSent>              <isRead>false</isRead>              <isEdited>false</isEdited>              <isDeleted>false</isDeleted>          </message>          <message>              <messageId>106003</messageId>              <chatId>103001</chatId>              <userId>101002</userId>              <text>Всё хорошо</text>              <mediaUrls>              </mediaUrls>              <timeSent>2025-10-11T09:10:00</timeSent>              <isRead>false</isRead>              <isEdited>false</isEdited>              <isDeleted>false</isDeleted>          </message>          <message>              <messageId>106004</messageId>              <chatId>103002</chatId>              <userId>101003</userId>              <text>А как лабу делать?</text>              <mediaUrls>              </mediaUrls>              <timeSent>2025-10-11T13:00:00</timeSent>              <isRead>false</isRead>              <isEdited>false</isEdited>              <isDeleted>false</isDeleted>          </message>          <message>              <messageId>106005</messageId>              <chatId>103002</chatId>              <userId>101004</userId>              <text>Чд?</text>              <mediaUrls>              </mediaUrls>              <timeSent>2025-10-11T14:55:00</timeSent>              <isRead>false</isRead>              <isEdited>false</isEdited>              <isDeleted>false</isDeleted>          </message>      </messages> |

|  |
| --- |
| data.xml чат |
| <chats>          <chat>              <chatId>103001</chatId>              <usersId>                  <user>101001</user>                  <user>101002</user>                  <user>101003</user>                  <user>101004</user>              </usersId>              <createdAt>2025-10-01T12:00:00</createdAt>              <avatarUrl>                  https://static0.gamerantimages.com/wordpress/wp-content/uploads/2024/07/clash-royale-cards.jpg?q=49&amp;fit=crop&amp;w=825&amp;dpr=2</avatarUrl>              <title>Клеш Рояль</title>          </chat>          <chat>              <chatId>103002</chatId>              <usersId>                  <user>101002</user>                  <user>101003</user>              </usersId>              <createdAt>2025-10-05T18:30:00</createdAt>              <avatarUrl>https://i.pinimg.com/736x/80/64/4a/80644aeb5df244c0b4b722cfa1e0b47b.jpg</avatarUrl>              <title>Bestики</title>          </chat>          <chat>              <chatId>103003</chatId>              <usersId>                  <user>101001</user>                  <user>101004</user>              </usersId>              <createdAt>2025-10-07T09:00:00</createdAt>              <avatarUrl>                  https://avatars.mds.yandex.net/i?id=0168e824d01d542d52d9472033815d61\_sr-5859172-images-thumbs&amp;n=13</avatarUrl>              <title>Кабанчики</title>          </chat>      </chats> |

|  |
| --- |
| data.xml истории |
| <stories>          <story>              <storyId>110001</storyId>              <userId>101003</userId>              <description>Новая демка</description>              <mediaUrl>https://example.com/story1.mp4</mediaUrl>              <timePublish>2025-10-11T11:00:00</timePublish>              <viewersCount>1</viewersCount>              <reactions>                  <reaction>108001</reaction>              </reactions>              <viewers>                  <viewer>101001</viewer>              </viewers>              <isArchived>false</isArchived>              <isDeleted>false</isDeleted>          </story>          <story>              <storyId>110002</storyId>              <userId>101001</userId>              <description>Катка в CR</description>              <mediaUrl>https://example.com/story2.mp4</mediaUrl>              <timePublish>2025-10-11T12:10:00</timePublish>              <viewersCount>0</viewersCount>              <reactions>              </reactions>              <viewers>              </viewers>              <isArchived>false</isArchived>              <isDeleted>false</isDeleted>          </story>      </stories> |

|  |
| --- |
| data.xml уведомления |
| <notifications>          <notification>              <notificationId>107001</notificationId>              <userId>101001</userId>              <text>ВСем привет</text>              <clickUrl>https://cheburashki.net/chats/103001</clickUrl>              <timePublish>2025-10-11T08:30:00</timePublish>              <isRead>false</isRead>              <isOpened>false</isOpened>          </notification>          <notification>              <notificationId>107002</notificationId>              <userId>101003</userId>              <text>А как лабу делать?</text>              <clickUrl>https://cheburashki.net/chats/103002</clickUrl>              <timePublish>2025-10-11T13:00:00</timePublish>              <isRead>false</isRead>              <isOpened>false</isOpened>          </notification>      </notifications> |

|  |
| --- |
| data.xml товар на маркетплейс |
| <marketplaceItems>          <marketplaceItem>              <itemId>105001</itemId>              <title>Лабуба ориг</title>              <description>Оригинальная Лабуба прямо из Китая, досталась по наследству</description>              <photoUrls>                  <url>https://avatars.dzeninfra.ru/get-zen\_doc/271828/pub\_68e3d9bc98ee0c2fb7eee9f0\_68e3d9cccbdb0445cf0b2cbb/scale\_1200</url>              </photoUrls>              <sellerId>101001</sellerId>              <price>40000</price>              <status>active</status>              <createdAt>2025-03-10T00:00:00</createdAt>          </marketplaceItem>          <marketplaceItem>              <itemId>105002</itemId>              <title>Утюг</title>              <description>Старинный утюг времён варягов и греков</description>              <photoUrls>                  <url>https://avatars.mds.yandex.net/i?id=ddc8952e84a2e1f0cdf1780484390556\_l-3524219-images-thumbs&amp;n=13</url>              </photoUrls>              <sellerId>101001</sellerId>              <price>300</price>              <status>active</status>              <createdAt>2023-09-30T00:00:00</createdAt>          </marketplaceItem>          <marketplaceItem>              <itemId>105003</itemId>              <title>Велосипед</title>              <description>Алюминиевый велосипед Stels</description>              <photoUrls>                  <url>https://avatars.mds.yandex.net/get-mpic/5335973/2a00000191e501bdb77f5948b454b8bd9950/9hq</url>              </photoUrls>              <sellerId>101004</sellerId>              <price>10000</price>              <status>sold</status>              <createdAt>2024-06-15T00:00:00</createdAt>          </marketplaceItem>      </marketplaceItems> |

# Вывод

В данной лабораторной работе научились писать читаемый и понятный код в стиле ООП. Для примера были созданы классы для области Социальная сеть. На её примере мы научились создавать классы с понятной аннотацией, методами CRUD и обрабатывать ошибки в классах, а также записывать данные в json и считывать их. Для этого использовали сериализацию и десериализацию данных. В ходе работы также были применены навыки дизайна для создания UML-диаграмм и продумывание классов и методов перед написанием кода.

# Ссылка на удаленный репозиторий

https://github.com/67Konstantin/labs\_app\_programming/tree/lab1/lab1