Цель работы

Изучить базовые понятия онтологического подхода и инструментальные средства онтологического проектирования, а также получить навыки работы с редактором онтологий Owlready2.

Выбранная предметная область: Работа музыкального стримингового сервиса.

Лабораторное задание

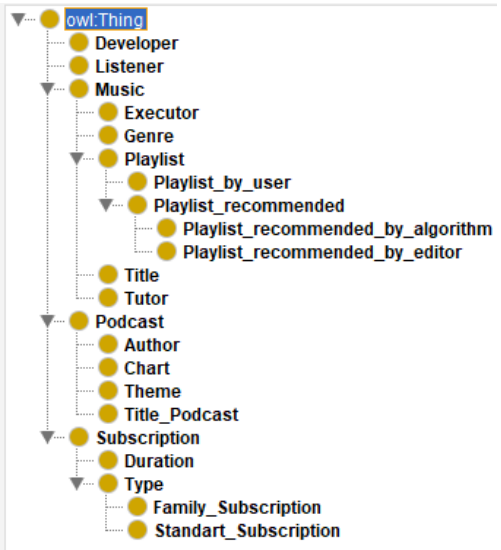
1. Для выбранной предметной области выделить 10-20 понятий (концептов).

Выбранная предметная область: Работа музыкального стримингового сервиса.

2. Дать определения этим понятиям.

Результаты проектирования

Protege



Python

```
Список классов онтологии:
onto.Music
onto.Title
onto.Genre
onto.Executor
onto.Tutor
onto.Playlist
onto.Playlist_by_user
onto.Playlist_recommended
onto.Playlist_recommended_by_algorithm
onto.Playlist_recommended_by_editor
onto.Podcast
onto.Title_Podcast
onto.Author
onto.Theme
onto.Chart
onto.Subscription
onto.Duration
onto.Type
onto.Family_Subscription
onto.Standart_Subscription
onto.Developer
onto.Listener
```

Код проектирования классов сущностей:

```
with onto:
    # музыка
    class Music(owlready2.Thing):
        pass
```

```
class Title(Music):
    pass
class Genre(Music):
    pass
class Executor(Music):
    pass
class Tutor(Music):
    pass

class Playlist(Music):
    pass
class Playlist_by_user(Playlist):
    pass
class Playlist_recommended(Playlist):
    pass
class Playlist_recommended_by_algorithm(Playlist_recommended):
    pass
class Playlist_recommended_by_editor(Playlist_recommended):
    pass


#подкаст
class Podcast(owlready2.Thing):
        pass
class Title_Podcast(Podcast):
    pass
class Author(Podcast):
    pass
class Theme(Podcast):
    pass
class Chart(Podcast):
    pass

#подписка
class Subscription(owlready2.Thing):
        pass
class Duration(Subscription): #продолжительность подписки
        pass

class Type(Subscription):
        pass
class Family_Subscription(Type):
        pass
class Standart_Subscription(Type):
        pass

#разработчик
class Developer(owlready2.Thing):
        pass

#слушатель
class Listener(owlready2.Thing):
        pass
```

Понятия и их определения:

1. Музыка — вид искусства, в котором определенным образом организованные музыкальные звуки используются для создания

некоторого сочетания формы, гармонии, мелодии, ритма или иного выразительного содержания.

1. Название произведения - название любой музыкальной композиции, представленной в данном стриминговом сервисе.

2. Жанр (Музыкальный жанр) — род музыки, музыкальных произведений, характеризующийся определёнными сюжетными, композиционными, стилистическими и др. признаками; а также отдельные разновидности этого рода.

3. Исполнитель — физическое лицо, творческим трудом которого осуществлено исполнение: артист-исполнитель (актер, певец, музыкант, танцор или другое лицо, которое играет роль, читает, декламирует, поет, играет на музыкальном инструменте или иным образом участвует в исполнении произведения литературы, искусства)

4. Куратор — тематическое сообщество, которое превратилось в экспертную редакцию. Кураторам регулярно получается заинтересовать аудиторию: авторскими плейлистами, мнением о событиях в музыкальной индустрии.

5. Плейлист — список воспроизведения аудиоконтента

    5. 1. Плейлист, собранный пользователем

    5. 2. Плейлист, рекомендуемый пользователю

        5.2.1 Плейлист, собранный алгоритмами - плейлист, созданный с использованием ИИ, алгоритмов подбора песен

        5.2.2 Плейлист, собранный редакцией - плейлист, собранный определенным редактором, музыкантом.

2. Подкаст – это аудиопрограмма, которую можно слушать онлайн в приложениях на телефоне или скачивать и слушать тогда, когда у вас нет подключения к интернету.

    1. Название подкаста — уникальное имя подкаста в системе.

2. Автор — физическое лицо, творческим трудом которого создан подкаст.

3. Тема подкаста — центр произведения; то, о чём говорит автор. Особенности: тема раскрывается автором в совокупности явлений и событий, которые он описывает и оценивает. Бывают развлекательные, новостные, нарративные, образовательные, подкасты-расследования и др.

3. Подписка — это бизнес-модель, при которой клиент должен регулярно оплачивать повторяющуюся сумму за доступ к товару или услуге.

   1. Продолжительность подписки — период, на который будет осуществлена подписка.

   2. Тип подписки. Различают семейную и стандартную подписку. Стандартная подписка распространяется на один аккаунт пользователя. Семейная подписка — это опция для всей семьи по выгодной цене.

4. Разработчик-редактор — специалист, который занимаемся созданием стримингового сервиса.

5. Слушатель — пользователь, оформивший подписку на стриминговый музыкальный сервис.

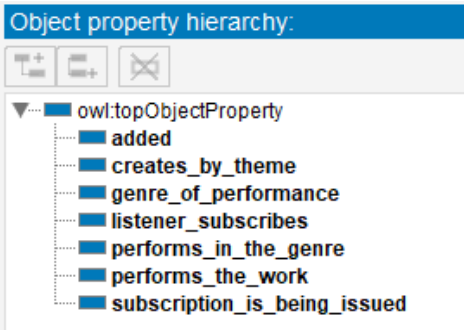
3. На множестве понятий ввести отношения и функции интерпретации для построения онтологии по предметной области.

Свойства объектов	**Обратные к ним**
жанр_выступления (*Asymmetric, Irreflexive*) Жанр -> Исполнитель	**выступает_в** (*Asymmetric, Irreflexive*) Исполнитель -> Жанр
добавил (*Symmetric*) Разработчик-редактор -> Плейлист, созданный редакцией (Плейлист, созданный редакцией-> Разработчик-редактор)	
исполняет (*Irreflexive*) Исполнитель -> Название произведения	

оформляет *(Functional, Inverse of* **оформляется***)* Слушатель ->Подписка	**оформляется** *(Functional, Inverse of* **оформляет***)* Подписка ->Слушатель
создает_по_теме (*Irreflexive*) Автор-> Тема подкаста	

Список свойств онтологии

Protege



Python

```
Список свойств онтологии:
onto.performs_in_the_genre
onto.genre_of_performance
onto.added
onto.performs_the_work
onto.listener_subscribes
onto.subscription_is_being_issued
onto.creates_by_theme
```

Код создания свойств:

```
# выступает в жанре
class performs_in_the_genre(Executor >> Genre, owlready2.AsymmetricProperty,
owlready2.IrreflexiveProperty):
   pass
# жанр выступления
class genre_of_performance (Genre >> Executor, owlready2.AsymmetricProperty,
owlready2.IrreflexiveProperty):
   pass
# добавил
class added (Genre >> Playlist, owlready2.SymmetricProperty): #
   pass
# исполняет произведение
class performs_the_work (Executor >> Title , owlready2.AsymmetricProperty,
owlready2.IrreflexiveProperty):
   pass

#слушатель оформляет подписку (причем 1 слушатель может оформить 1 подписку, функциональное св-во)
class listener_subscribes(owlready2.ObjectProperty,owlready2.FunctionalProperty):
   domain = [Listener]
   range = [Subscription]

# подписка оформляется слушателем (обратное свойство к "слушатель оформляет подписку ")
class subscription_is_being_issued(owlready2.ObjectProperty):
   domain = [Subscription]
   range =  [Listener]
   inverse_property = listener_subscribes
```

```
# создает подкаст по теме
class creates_by_theme (Author>> Theme, owlready2.AsymmetricProperty, owlready2.IrreflexiveProperty):
   pass
```

Создание функциональных свойств, имеющих единственное значение для данного экземпляра. Функциональные свойства создаются путем наследования класса FunctionalProperty.

```
 # почта слушателя
 class Listener_mail(owlready2.DataProperty, owlready2.FunctionalProperty):
    domain = [Listener]
    range = [str]

 # идентификатор слушателя в БД сервиса
 class Listener_ID(owlready2.DataProperty, owlready2.FunctionalProperty):
    domain = [Listener]
    range = [int]

 # идентификатор исполнителя в БД сервиса
 class Executor_ID(owlready2.DataProperty, owlready2.FunctionalProperty):
    domain = [Executor]
    range = [int]

 # прибыль исполнителя
 class Executor_profit(owlready2.DataProperty, owlready2.FunctionalProperty):
    domain = [Executor]
    range = [float]

 # прибыль автора подкаста
 class Author_profit(owlready2.DataProperty, owlready2.FunctionalProperty):
    domain = [Author]
    range = [float]
```

4. Создать экземпляры классов

```
# экземпляры класса
jazz = Genre ("Джаз")
classical_music = Genre ("Классическая музыка")
pop = Genre ("Поп")
rock = Genre ("Рок")
hip_hop = Genre ("Хип-хоп")
electronic_music = Genre ("Электронная музыка")
rap = Genre ("Рэп")
# исполнители музыки
Egor_Krid = Executor("Егор Крид", Executor_ID = 1, Executor_profit = 1000)
Maxim = Executor("Мак$им", Executor_ID = 2, Executor_profit = 2000)
Amy_Winehouse = Executor("Amy Winehouse", Executor_ID = 3, Executor_profit = 3000)
Arctic_Monkeys = Executor("Arctic Monkeys", Executor_ID = 4, Executor_profit = 4000)
Lana_del_Rey = Executor("Lana del Rey", Executor_ID = 5, Executor_profit = 5000)
Ludovico_Einaudi = Executor("Ludovico Einaudi", Executor_ID = 6, Executor_profit = 6000)
Hans_Zimmer = Executor("Hans Zimmer", Executor_ID = 7, Executor_profit = 7000)
Daft_Punk = Executor("Daft Punk", Executor_ID = 8, Executor_profit = 8000)
Eminem = Executor("Eminem", Executor_ID = 9, Executor_profit = 9000)
# авторы подкастов
Mamenko = Author("Маменко", Author_profit = 5000)
Yuzefovich= Author("Юзефович", Author_profit = 7000)
# темы подкастов
Humor = Theme("Юмор")
Science_and_education= Theme("Наука и образование")
# слушатели
```

```
Abramov = Listener("Абрамов", Listener_ID = 1, Listener_mail = "abramov2000@mail.ru")
Bychkov = Listener("Бычков", Listener_ID = 2, Listener_mail = "bychkov@mail.ru")
Vasnetsova = Listener("Васнецова", Listener_ID = 3, Listener_mail = "fan_of_Sergey_Lazarev@mail.ru")
Ryan_Goslingova = Listener("Райан-Гослингова", Listener_ID = 4, Listener_mail =
"real_ryan_goslings_wife@mail.ru")
# разработчики
Razrabotchikov = Developer("Разработчиков")
Yandexov = Developer("Яндексов")
# плейлисты
Autumn_playlist = Playlist_recommended_by_editor("Осенний плейлист")
Tik_tok_top = Playlist_recommended_by_editor("Тик-ток топ")
# тип подписки
standart_1_month = Subscription("Стандартная подписка на 1 месяц")
family_1_month = Family_Subscription("Семейная подписка на 1 месяц")


Egor_Krid = Executor("Егор Крид", Executor_ID = 1, Executor_profit = 1000)
Maxim = Executor("Мак$им", Executor_ID = 2, Executor_profit = 2000)
Amy_Winehouse = Executor("Amy Winehouse", Executor_ID = 3, Executor_profit = 3000)
Arctic_Monkeys = Executor("Arctic Monkeys", Executor_ID = 4, Executor_profit = 4000)
Lana_del_Rey = Executor("Lana del Rey", Executor_ID = 5, Executor_profit = 5000)
Ludovico_Einaudi = Executor("Ludovico Einaudi", Executor_ID = 6, Executor_profit = 6000)
Hans_Zimmer = Executor("Hans Zimmer", Executor_ID = 7, Executor_profit = 7000)
Daft_Punk = Executor("Daft Punk", Executor_ID = 8, Executor_profit = 8000)
Eminem = Executor("Eminem", Executor_ID = 9, Executor_profit = 9000)

Otpyskay = Title("Отпускаю")
October_song = Title("October_song")
Rap_god = Title ("Rap god")
Fluorescent_Adolescent = Title ("Fluorescent Adolescent")
Dark_Paradise = Title ("Dark_Paradise")
Radio = Title ("Radio")
Experience = Title ("Experience")
STAY = Title ("S.T.A.Y.")
Get_lucky = Title ("Get_lucky")

# жанр выступления
jazz.genre_of_performance = [Amy_Winehouse]
classical_music.genre_of_performance = [Ludovico_Einaudi, Hans_Zimmer]
pop.genre_of_performance = [Egor_Krid, Lana_del_Rey, Maxim]
rock.genre_of_performance = [Arctic_Monkeys, Lana_del_Rey]
electronic_music.genre_of_performance = [Daft_Punk]
rap.genre_of_performance = [Eminem]

# выступает в
Egor_Krid.performs_in_the_genre =[pop, rap]
Maxim.performs_in_the_genre =[pop]
Amy_Winehouse.performs_in_the_genre =[jazz]
Arctic_Monkeys.performs_in_the_genre =[rock]
Lana_del_Rey.performs_in_the_genre =[pop, rock]
Ludovico_Einaudi.performs_in_the_genre =[classical_music]
Hans_Zimmer.performs_in_the_genre =[classical_music]
Daft_Punk.performs_in_the_genre =[electronic_music]
Eminem.performs_in_the_genre =[rap]

# добавил (симметричное)
Razrabotchikov.added = [Autumn_playlist]
Yandexov.added = [Tik_tok_top]

# исполняет
Egor_Krid.performs_the_work = [Otpyskay]
Maxim.performs_the_work = [Otpyskay]
Amy_Winehouse.performs_the_work = [October_song]
```

```
Arctic_Monkeys.performs_the_work = [Fluorescent_Adolescent]
Lana_del_Rey.performs_the_work = [Dark_Paradise, Radio]
Ludovico_Einaudi.performs_the_work = [Experience]
Hans_Zimmer.performs_the_work = [STAY]
Daft_Punk.performs_the_work = [Get_lucky]
Eminem.performs_the_work = [Rap_god]

# оформляет
Abramov.listener_subscribes = standart_1_month
Bychkov.listener_subscribes = standart_1_month
Vasnetsova.listener_subscribes =standart_1_month
Ryan_Goslingova.listener_subscribes = family_1_month

# оформляется - обратно к оформляет

# создает по теме
Mamenko.creates_by_theme = [Humor]
Yuzefovich.creates_by_theme = [Science_and_education]
```

Список экземпляров

onto.Джаз
onto.Классическая музыка
onto.Поп
onto.Рок
onto.Хип-хоп
onto.Электронная музыка
onto.Рэп
onto.Егор Крид
onto.Мак$им
onto.Amy Winehouse
onto.Arctic Monkeys
onto.Lana del Rey
onto.Ludovico Einaudi
onto.Hans Zimmer
onto.Daft Punk
onto.Eminem
onto.Маменко
onto.Юзефович
onto.Юмор
onto.Наука и образование
onto.Абрамов
onto.Бычков
onto.Васнецова
onto.Райан-Гослингова
onto.Разработчиков
onto.Яндексов
onto.Осенний плейлист
onto.Тик-ток топ
onto.Стандартная подписка на 1 месяц
onto.Семейная подписка на 1 месяц
onto.Отпускаю
onto.October_song
onto.Rap god
onto.Fluorescent Adolescent
onto.Dark_Paradise
onto.Radio
onto.Experience
onto.S.T.A.Y.
onto.Get_lucky

5. Осуществить поиск информации по разработанной предметной онтологии.

Запросы:

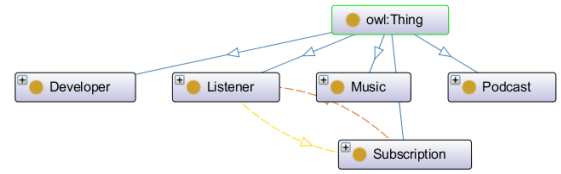
```
# жанр выступления
print("1. Жанр выступления Ханса Циммера:", onto.search(genre_of_performance = Hans_Zimmer))
print("2. Жанр выступления Daft Punk:", onto.search(genre_of_performance = Daft_Punk))
# выступает в
print("3. Кто выступает в жанре поп с прибылью 1000:", onto.search(performs_in_the_genre = pop,
Executor_profit = 1000))
print("4. Кто выступает в жанре поп:", onto.search(performs_in_the_genre = pop))
# добавил(симметричное)
print("5. Добавил осенний плейлист:", onto.search(added= Autumn_playlist))
print("6. Тик-ток топ добавил: ", onto.search(added = Tik_tok_top))
# исполняет
print("7. Кто исполняет S.T.A.Y.: ", onto.search(performs_the_work = STAY))
print("8. Кто исполняет Отпускаю: ", onto.search(performs_the_work = Otpyskay))
# оформляет
print("9. Кто оформляет стандартную подписку на месяц: ", onto.search(listener_subscribes =
standart_1_month))
print("10. Кто оформляет семейную подписку на месяц: ", onto.search(listener_subscribes =
family_1_month))
# оформляется
print("11. Какую подписку оформляет Райан-Гослингова: ", onto.search(subscription_is_being_issued =
Ryan_Goslingova))
print("12. Какую подписку оформляет Разработчиков: ", onto.search(subscription_is_being_issued =
Bychkov))
# создает по теме
print("13. Кто создает подкасты по теме юмор: ", onto.search(creates_by_theme = Humor))
print("14. Кто создает подкасты по теме наука и образование: ", onto.search(creates_by_theme =
Science_and_education))
```

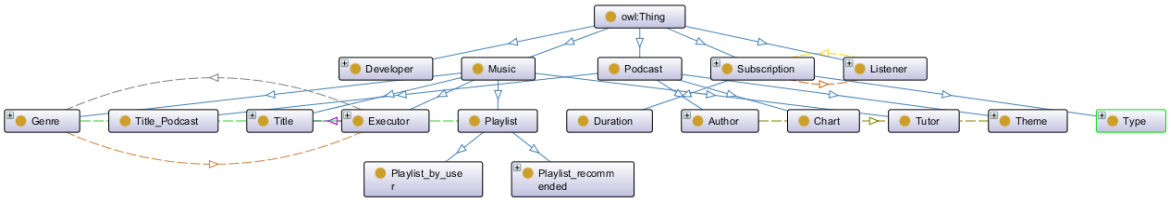
Результат:

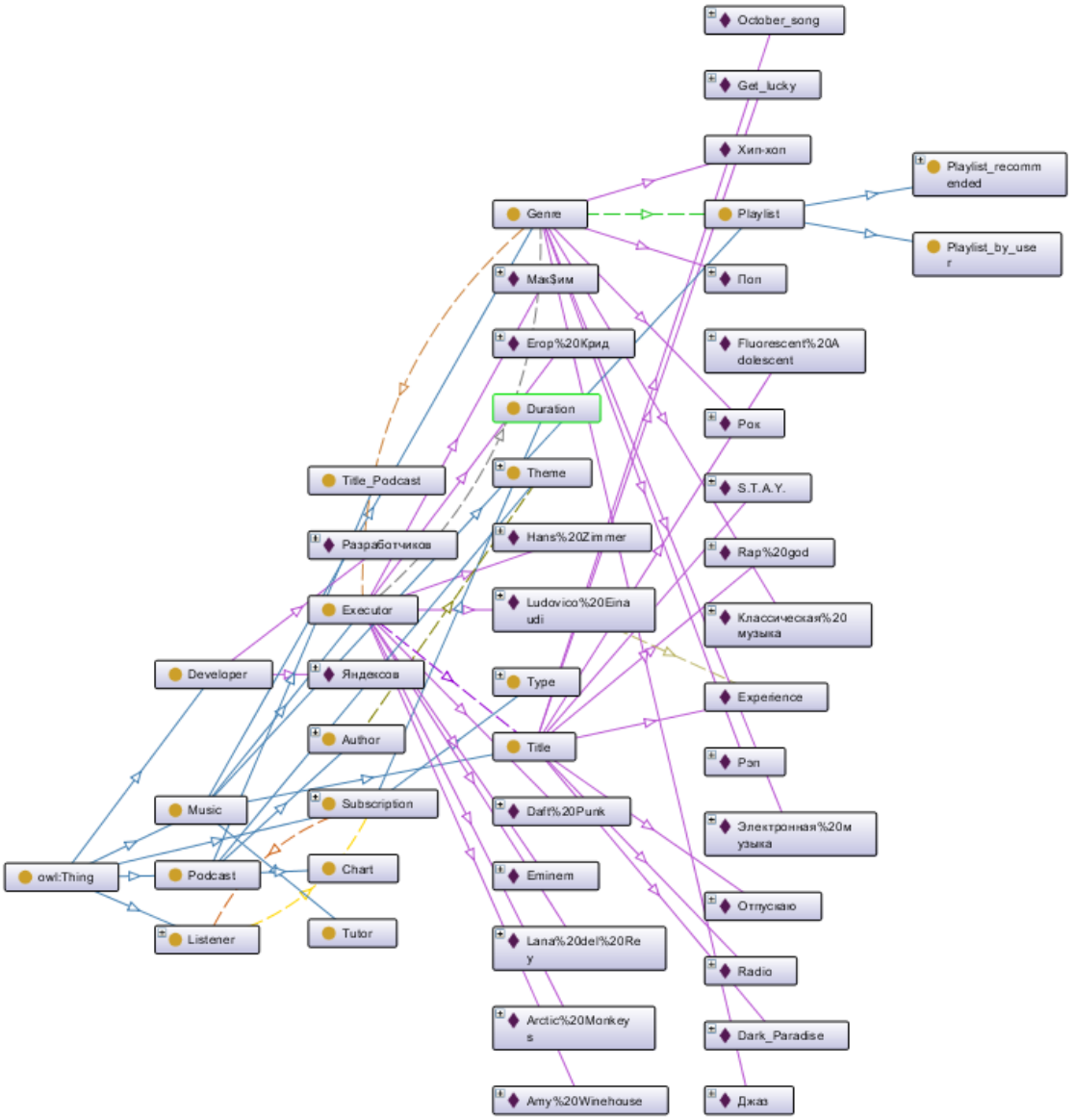
```
1. Жанр выступления Ханса Циммера: [onto.Классическая музыка]
2. Жанр выступления Daft Punk: [onto.Электронная музыка]
3. Кто выступает в жанре поп с прибылью 1000: [onto.Егор Крид]
4. Кто выступает в жанре поп: [onto.Егор Крид, onto.Мак$им, onto.Lana del Rey]
5. Добавил осенний плейлист: [onto.Разработчиков]
6. Тик-ток топ добавил:  [onto.Яндексов]
7. Кто исполняет S.T.A.Y.:  [onto.Hans Zimmer]
8. Кто исполняет Отпускаю:  [onto.Егор Крид, onto.Мак$им]
9. Кто оформляет стандартную подписку на месяц:  [onto.Абрамов, onto.Бычков, onto.Васнецова]
10. Кто оформляет семейную подписку на месяц:  [onto.Райан-Гослингова]
11. Какую подписку оформляет Райан-Гослингова:  [onto.Семейная подписка на 1 месяц]
12. Какую подписку оформляет Разработчиков:  [onto.Стандартная подписка на 1 месяц]
13. Кто создает подкасты по теме юмор:  [onto.Маменко]
14. Кто создает подкасты по теме наука и образование:  [onto.Юзефович]

Process finished with exit code 0
```

Визуализация онтологии







Заключение

В рамках данной работы была построена онтология, отражающая предметную область, изучены основы работы с библиотекой owlready2, а также были изучены возможности поиска по онтологии, показано, что использование языка Python с библиотекой owlready2 позволяет автоматизировать получение результатов, за счет использования циклов.

Литература

1. Документация Owlready2

Программный код

```
from email.header import Header
from sqlite3 import ProgrammingError
import owlready2

owl_path_new = r"http://test.org/onto.owl"
onto = owlready2.get_ontology(owl_path_new)

with onto:
    #--------------------
    # экземпляры класса
    # музыка
    class Music(owlready2.Thing):
        pass
    class Title(Music):
        pass
    class Genre(Music):
        pass
    class Executor(Music):
        pass
    class Tutor(Music):
        pass

    class Playlist(Music):
        pass
    class Playlist_by_user(Playlist):
        pass
    class Playlist_recommended(Playlist):
        pass
    class Playlist_recommended_by_algorithm(Playlist_recommended):
        pass
    class Playlist_recommended_by_editor(Playlist_recommended):
        pass


    #подкаст
    class Podcast(owlready2.Thing):
            pass
    class Title_Podcast(Podcast):
        pass
    class Author(Podcast):
        pass
    class Theme(Podcast):
        pass
    class Chart(Podcast):
        pass

    #подписка
    class Subscription(owlready2.Thing):
            pass
    class Duration(Subscription): #продолжительность подписки
            pass

    class Type(Subscription):
            pass
    class Family_Subscription(Type):
            pass
    class Standart_Subscription(Type):
            pass

    #разработчик
    class Developer(owlready2.Thing):
```

```
            pass

    #слушатель
    class Listener(owlready2.Thing):
            pass

    #-------------------
    # Свойства
    # выступает в жанре
    class performs_in_the_genre(Executor >> Genre, owlready2.AsymmetricProperty,
owlready2.IrreflexiveProperty):
        pass
    # жанр выступления
    class genre_of_performance (Genre >> Executor, owlready2.AsymmetricProperty,
owlready2.IrreflexiveProperty):
        pass
    # добавил
    class added (Genre >> Playlist, owlready2.SymmetricProperty): #
        pass
    # исполняет произведение
    class performs_the_work (Executor >> Title , owlready2.AsymmetricProperty,
owlready2.IrreflexiveProperty):
        pass

    #слушатель оформляет подписку (причем 1 слушатель может оформить 1 подписку, функциональное
св-во)
    class listener_subscribes(owlready2.ObjectProperty,owlready2.FunctionalProperty):
        domain = [Listener]
        range = [Subscription]

    # подписка оформляется слушателем (обратное свойство к "слушатель оформляет подписку ")
    class subscription_is_being_issued(owlready2.ObjectProperty):
        domain = [Subscription]
        range =  [Listener]
        inverse_property = listener_subscribes

    # создает подкаст по теме
    class creates_by_theme (Author>> Theme, owlready2.AsymmetricProperty,
owlready2.IrreflexiveProperty):
        pass
    #-------------------
    # свойства данных
    # почта слушателя
    class Listener_mail(owlready2.DataProperty, owlready2.FunctionalProperty):
        domain = [Listener]
        range = [str]

    # идентификатор слушателя в БД сервиса
    class Listener_ID(owlready2.DataProperty, owlready2.FunctionalProperty):
        domain = [Listener]
        range = [int]

    # идентификатор исполнителя в БД сервиса
    class Executor_ID(owlready2.DataProperty, owlready2.FunctionalProperty):
        domain = [Executor]
        range = [int]

    # прибыль исполнителя
    class Executor_profit(owlready2.DataProperty, owlready2.FunctionalProperty):
        domain = [Executor]
        range = [float]

    # прибыль автора подкаста
```

```
    class Author_profit(owlready2.DataProperty, owlready2.FunctionalProperty):
        domain = [Author]
        range = [float]



#-------------------
# экземпляры класса
jazz = Genre ("Джаз")
classical_music = Genre ("Классическая музыка")
pop = Genre ("Поп")
rock = Genre ("Рок")
hip_hop = Genre ("Хип-хоп")
electronic_music = Genre ("Электронная музыка")
rap = Genre ("Рэп")
# исполнители музыки
Egor_Krid = Executor("Егор Крид", Executor_ID = 1, Executor_profit = 1000)
Maxim = Executor("Мак$им", Executor_ID = 2, Executor_profit = 2000)
Amy_Winehouse = Executor("Amy Winehouse", Executor_ID = 3, Executor_profit = 3000)
Arctic_Monkeys = Executor("Arctic Monkeys", Executor_ID = 4, Executor_profit = 4000)
Lana_del_Rey = Executor("Lana del Rey", Executor_ID = 5, Executor_profit = 5000)
Ludovico_Einaudi = Executor("Ludovico Einaudi", Executor_ID = 6, Executor_profit = 6000)
Hans_Zimmer = Executor("Hans Zimmer", Executor_ID = 7, Executor_profit = 7000)
Daft_Punk = Executor("Daft Punk", Executor_ID = 8, Executor_profit = 8000)
Eminem = Executor("Eminem", Executor_ID = 9, Executor_profit = 9000)
# авторы подкастов
Mamenko = Author("Маменко", Author_profit = 5000)
Yuzefovich= Author("Юзефович", Author_profit = 7000)
# темы подкастов
Humor = Theme("Юмор")
Science_and_education= Theme("Наука и образование")
# слушатели
Abramov = Listener("Абрамов", Listener_ID = 1, Listener_mail = "abramov2000@mail.ru")
Bychkov = Listener("Бычков", Listener_ID = 2, Listener_mail = "bychkov@mail.ru")
Vasnetsova = Listener("Васнецова", Listener_ID = 3, Listener_mail = "fan_of_Sergey_Lazarev@mail.ru")
Ryan_Goslingova = Listener("Райан-Гослингова", Listener_ID = 4, Listener_mail =
"real_ryan_goslings_wife@mail.ru")
# разработчики
Razrabotchikov = Developer("Разработчиков")
Yandexov = Developer("Яндексов")
# плейлисты
Autumn_playlist = Playlist_recommended_by_editor("Осенний плейлист")
Tik_tok_top = Playlist_recommended_by_editor("Тик-ток топ")
# тип подписки
standart_1_month = Subscription("Стандартная подписка на 1 месяц")
family_1_month = Family_Subscription("Семейная подписка на 1 месяц")


Egor_Krid = Executor("Егор Крид", Executor_ID = 1, Executor_profit = 1000)
Maxim = Executor("Мак$им", Executor_ID = 2, Executor_profit = 2000)
Amy_Winehouse = Executor("Amy Winehouse", Executor_ID = 3, Executor_profit = 3000)
Arctic_Monkeys = Executor("Arctic Monkeys", Executor_ID = 4, Executor_profit = 4000)
Lana_del_Rey = Executor("Lana del Rey", Executor_ID = 5, Executor_profit = 5000)
Ludovico_Einaudi = Executor("Ludovico Einaudi", Executor_ID = 6, Executor_profit = 6000)
Hans_Zimmer = Executor("Hans Zimmer", Executor_ID = 7, Executor_profit = 7000)
Daft_Punk = Executor("Daft Punk", Executor_ID = 8, Executor_profit = 8000)
Eminem = Executor("Eminem", Executor_ID = 9, Executor_profit = 9000)

Otpyskay = Title("Отпускаю")
October_song = Title("October_song")
Rap_god = Title ("Rap god")
Fluorescent_Adolescent = Title ("Fluorescent Adolescent")
Dark_Paradise = Title ("Dark_Paradise")
```

```
Radio = Title ("Radio")
Experience = Title ("Experience")
STAY = Title ("S.T.A.Y.")
Get_lucky = Title ("Get_lucky")

# жанр выступления
jazz.genre_of_performance = [Amy_Winehouse]
classical_music.genre_of_performance = [Ludovico_Einaudi, Hans_Zimmer]
pop.genre_of_performance = [Egor_Krid, Lana_del_Rey, Maxim]
rock.genre_of_performance = [Arctic_Monkeys, Lana_del_Rey]
electronic_music.genre_of_performance = [Daft_Punk]
rap.genre_of_performance = [Eminem]

# выступает в
Egor_Krid.performs_in_the_genre =[pop, rap]
Maxim.performs_in_the_genre =[pop]
Amy_Winehouse.performs_in_the_genre =[jazz]
Arctic_Monkeys.performs_in_the_genre =[rock]
Lana_del_Rey.performs_in_the_genre =[pop, rock]
Ludovico_Einaudi.performs_in_the_genre =[classical_music]
Hans_Zimmer.performs_in_the_genre =[classical_music]
Daft_Punk.performs_in_the_genre =[electronic_music]
Eminem.performs_in_the_genre =[rap]

# добавил (симметричное)
Razrabotchikov.added = [Autumn_playlist]
Yandexov.added = [Tik_tok_top]

# исполняет
Egor_Krid.performs_the_work = [Otpyskay]
Maxim.performs_the_work = [Otpyskay]
Amy_Winehouse.performs_the_work = [October_song]
Arctic_Monkeys.performs_the_work = [Fluorescent_Adolescent]
Lana_del_Rey.performs_the_work = [Dark_Paradise, Radio]
Ludovico_Einaudi.performs_the_work = [Experience]
Hans_Zimmer.performs_the_work = [STAY]
Daft_Punk.performs_the_work = [Get_lucky]
Eminem.performs_the_work = [Rap_god]

# оформляет
Abramov.listener_subscribes = standart_1_month
Bychkov.listener_subscribes = standart_1_month
Vasnetsova.listener_subscribes =standart_1_month
Ryan_Goslingova.listener_subscribes = family_1_month

# оформляется - обратно к оформляет

# создает по теме
Mamenko.creates_by_theme = [Humor]
Yuzefovich.creates_by_theme = [Science_and_education]

#-------------------
# запросы
# жанр выступления
print("1. Жанр выступления Ханса Циммера:", onto.search(genre_of_performance = Hans_Zimmer))
print("2. Жанр выступления Daft Punk:", onto.search(genre_of_performance = Daft_Punk))
# выступает в
print("3. Кто выступает в жанре поп с прибылью 1000:", onto.search(performs_in_the_genre = pop,
Executor_profit = 1000))
print("4. Кто выступает в жанре поп:", onto.search(performs_in_the_genre = pop))
# добавил(симметричное)
print("5. Добавил осенний плейлист:", onto.search(added= Autumn_playlist))
print("6. Тик-ток топ добавил: ", onto.search(added = Tik_tok_top))
```

```
# исполняет
print("7. Кто исполняет S.T.A.Y.: ", onto.search(performs_the_work = STAY))
print("8. Кто исполняет Отпускаю: ", onto.search(performs_the_work = Otpyskay))
# оформляет
print("9. Кто оформляет стандартную подписку на месяц: ", onto.search(listener_subscribes =
standart_1_month))
print("10. Кто оформляет семейную подписку на месяц: ", onto.search(listener_subscribes =
family_1_month))
# оформляется
print("11. Какую подписку оформляет Райан-Гослингова: ", onto.search(subscription_is_being_issued =
Ryan_Goslingova))
print("12. Какую подписку оформляет Разработчиков: ", onto.search(subscription_is_being_issued =
Bychkov))
# создает по теме
print("13. Кто создает подкасты по теме юмор: ", onto.search(creates_by_theme = Humor))
print("14. Кто создает подкасты по теме наука и образование: ", onto.search(creates_by_theme =
Science_and_education))

onto.save(file = "Abrosimova_LR_2.owl")
print("")

print("Список классов онтологии:")
for item in list(onto.classes()):
   print(item)

print("")
print("Список свойств онтологии:")
for item in list(onto.properties()):
   print(item)

print("")
print("Список экземпляров онтологии:")
for item in list(onto.individuals()):
   print(item)
```