ГБОУ Школа №1533 «ЛИТ»

Визуализация географических объектов на материале лингвистических корпусов (по текстам и метаданным)

Заказчик: Александр Владимирович Архипов,
доцент кафедры ТиПЛ (теоретической и
прикладной лингвистики) МГУ

Консультант: Николай Константинович Завриев

Исполнители: Потапова Дарья,
Леоненкова Виктория,
группа 10.4

Москва – 2020

Оглавление

Область проекта

Современная лингвистика представляет собой многогранную науку, активно использующуюся в самых разных областях, так как она связана со множеством разных наук (например, история, семантика, информатика, математика, философия, физика, биология и физиология). Актуальность лингвистических исследований связана с тем, что в настоящее время требуется обработка огромного количества текстовой информации, что и предполагает анализ этого материала, в том числе и с лингвистической точки зрения. Сейчас лингвистика пытается максимально связать изучение языка с компьютерными технологиями. И она в этом преуспевает. Новые технологические возможности создают новые источники данных о языке, что выражается в увеличении роли, которую играют экспериментальные и корпусные данные. В связи с этим появляются и новые области этой науки, такие как компьютерная лингвистика и корпусная лингвистика.

Корпусная лингвистика, в первую очередь, занимается разработкой, созданием и использованием текстовых, или лингвистических, корпусов с помощью компьютерных технологий. Такой корпус представляет множество подобранных и обработанных текстов на некотором языке, которые в дальнейшем используются в качестве базы при исследовании языка. С помощью корпуса можно узнать о структуре, грамматике языка, о лексических значениях единиц языка и их возможных использованиях в разных ситуациях. Самым популярным и самым ценным корпусом является так называемый «национальный корпус», в котором собрано максимальное количество всех типов текстов, имеющихся на данном языке. Например, в России таким корпусом является Национальный корпус русского языка.

Такие корпусы довольно часто используются в отношении к малым языкам, то есть к языкам, носителей которых достаточно небольшое количество. В нашем проекте мы использовали 3 текстовых корпуса трех малых языков России: камасинский, селькупский и долганский.

Чтобы отследить правильность выполнения программы, точность отображения точек на карте необходимо затронуть и лингвистическую географию – область лингвистики, которая занимается изучением территориального расположения языков и распространения языковых явлений. Все три языка, с которыми мы работаем – языки народов, живущих в Таймырском Долгано-Ненецком районе (Краснодарский край). Знание территории их распространения значительно облегчает проверку результата на безошибочное выполнение.

Задача также имеет непосредственное отношение к геоинформационным системам, которые в современном мире пользуются большой популярностью, используются во многих сферах жизни и внедряются во множество приложений, поэтому программа, отображающая на карте места по материалам лингвистических корпусов, будет актуальна для людей, занимающихся или интересующихся лингвистической географией – областью лингвистики, которая занимается изучением территориального расположения языков и распространения языковых явлений.

Актуальность

В современном мире очень ценятся знания и информация. Этот проект актуален, так как позволяет извлекать информацию из языкового корпуса и преобразовывать её в такую форму, с которой будет удобно работать специалистам, в данном случае лингвистам.

В проекте осуществлено автоматическое извлечение необходимой информации из корпуса и текстов на данных языках, что дает возможность анализа изначально «хаотичной» информации с помощью стандартных методов обработки данных. На основе полученной информации, удобной для анализа, лингвисты имеют возможность наглядно и структурировано осуществлять исследование информации, выявлять логические закономерности, сохранять знания.

Лингвистическая информация и знания, составляют основу интеллектуального капитала нашей страны. Языки передают уникальность культуры, хранят человеческое наследие, накопленный объем знаний. Каждый язык уникален. Не существует двух одинаковых. Мы теряем древние знания, если теряем языки. Лингвисты, филологи стараются сохранить информацию о каждом языке и этот проект большая помощь в их работе, так как он позволяет извлечь информацию о языке из корпуса, структурировать ее, визуально и наглядно представить, сохранить знания.

Постановка задачи

Написать программу, отображающую на карте России места, связанные с несколькими малыми языками Российской Федерации, по материалам лингвистических корпусов текстов на этих языках. На карте можно будет увидеть населенные пункты, где живут (жили) говорящие; места, где были записаны те или иные тексты из корпуса; места, упоминаемые в текстах; и связи между рассказчиками, текстами и географическими объектами.

Целевая аудитория

Проект предназначен для лингвистов, исследующих малые языки России, и для более широкого круга пользователей, интересующихся малыми языками, историей и географией региона, например, краеведам.

Анализ предметной области

Основной ресурс – лингвистический корпус – множество подобранных и обработанных текстов, которые используются в качестве базы для исследования языка. Программа работает с 3 корпусами для 3 разных малых языков (т.е. языков, которыми пользуются немногочисленные, компактно проживающие народы): селькупский, камасинский, долганский.

Вся необходимая информация о корпусе или конкретном тексте из корпуса извлекается из метаданных (от лат. meta – цель, конечный пункт, предел, край и данные). Метаданные – «информация о другой информации», т.е. данные, которые относятся к дополнительной информации о содержимом. Они помогают раскрыть признаки и свойства объекта, упрощают управление, создание запросов, полноценное использование и понимание данных. Все метаданные для каждого из корпусов собраны в один общий файл XML (формат COMA). Каждый текст в корпусе хранится в отдельном файле XML (формат EXB).

XML (от англ. eXtensible Markup Language) – расширяемый язык разметки (т.е. набор специальных инструкций, называемых тэгами, предназначенных для формирования в документах какой-либо структуры и определения отношений между различными элементами этой структуры). XML также называют метаязыком, потому что он является очень удобным форматом представления метаданных и обмена ими, решает задачи доступа к ним.

Кроме этого, имеется файл KML с координатами и названиями, заполненный для поселков, где живут/жили говорящие на данных языках. KML (от англ. Keyhole Markup Language) – язык разметки на основе XML, используемый для отображения географических данных на карте. Для своей системы отсчета KML использует 3D-географические координаты: долготу, широту и высоту. Файла KML содержит карту с поселками, где собирались тексты для лингвистического корпуса. Если по каким-либо причинам в метаданных текста указано название места, но нет его географических

координат, они ищутся в файле KML и добавляются, чтобы потом подставлять автоматически (по возможности).

После отображения всех мест записи текстов идет работа с визуализацией географических данных, т.е. с геоинформационными системами (геоинформационные системы, или ГИС, – система сбора, хранения, анализа и визуализации пространственных данных и связанной с ними информации). Для этого из текстов необходимо извлечь все названия географических объектов. Это происходит путем парсинга (parsing) – принятого в информатике определения синтаксического анализа текста. Для парсинга создается некая математическая модель сравнения лексем (лексема – слово как абстрактная единица морфологического анализа, слово во всей совокупности его форм и значений) с формальной грамматикой, т.е. со словами, которые есть в «словарном запасе».

Обзор аналогов

Наш проект не имеет прямых аналогов, но удалось найти 2 похожие по реализации программы.

Первый аналог – функция PowerView, используемая в Excel для визуализации данных, с помощью которой создаются интерактивные карты, графики, диаграммы. Нас интересует именно создание карт. На карте PowerView есть визуализация в виде точек, при наведении на которую можно посмотреть информацию о данной метке. Также наличие поиска с помощью фильтров делает эту функцию немного похожей на наш проект. Но различия между нашей задачей и задачей PowerView существенное. Во-первых, наша цель – получить отдельное приложение с картой, а не встраиваемую функцию. Во-вторых, одна из главнейших задач нашего проекта – показать связи между метками на карте. Использование PowerView не предполагает такого. Кроме этого, эта функция свои данные берет не из текстового корпуса, а из таблиц Excel, что, конечно, является значительным различием между ней и нашим проектом.

Второй аналог, который имеет несколько схожий функционал с нашим проектом – интерактивный атлас UNESCO Atlas of the World’s Languages in danger. Основная задача этого атласа – визуализировать территориальное размещение исчезающих языков по всему миру. Все языки здесь делятся на 5 групп в зависимости от степени угрозы: от наименее уязвимых до полностью исчезнувших. Эта градация показана на карте точками разных цветов. При нажатии на маркер отображается информация об этом языке: название, страны распространения, количество носителей, географические координаты. Реализован удобный

поиск с помощью выпадающих списков и поисковых строк. В плане визуализации интерактивный атлас UNESCO очень похож на нашу программу, хотя, конечно, и имеет существенные различия в цели. Здесь, как и в PowerView, не предполагается визуализация каких-либо связей между точками. Этот атлас – интерактивная копия бумажного издания, последняя версия которого была издана в 2010 году, и все данные для карты берутся именно оттуда, а не из лингвистических корпусов или метаданных.

Программная реализация

Основная часть проекта, которая отвечает за работу с метаданными, текстовыми данными и файлами KML, написана на языке C#, как было согласовано с заказчиком. Среда разработки, в которой мы работали, – Microsoft Visual Studio 2019.

Для работы с файлами форматов .coma, .kml мы использовали текстовый редактор Sublime Text 3.

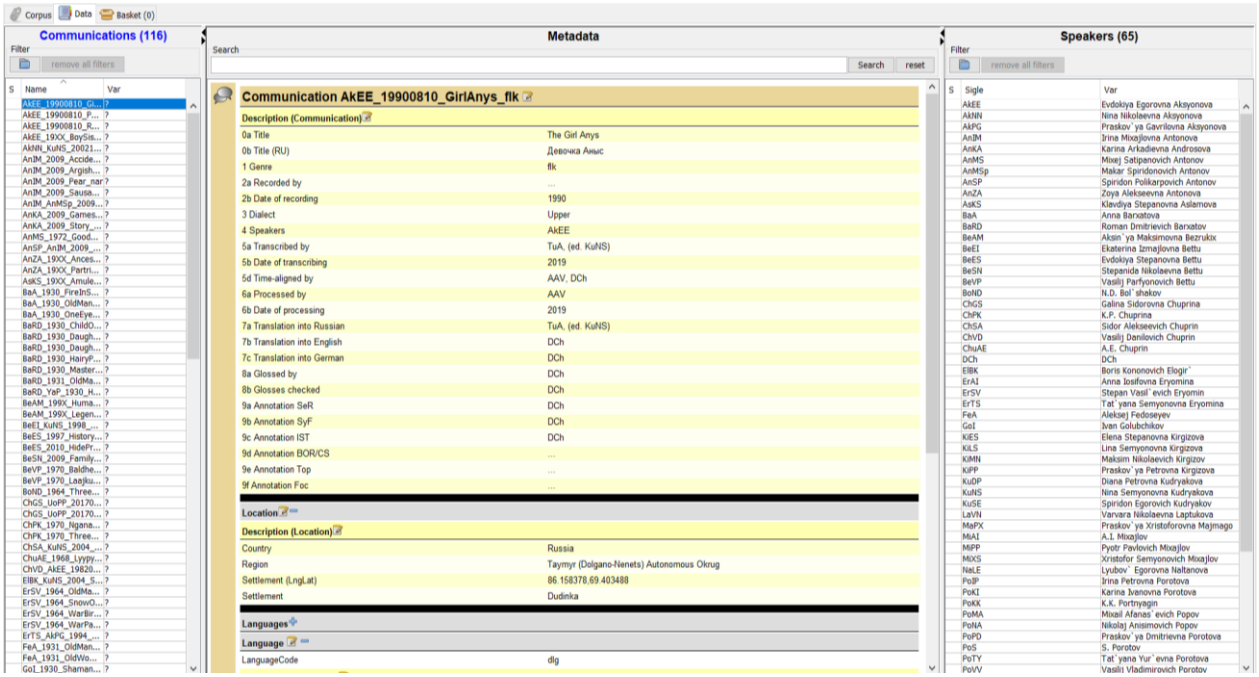
Также для работы с файлами формата .coma и .exb, мы пользовались пакетом EXMARaLDA, а именно программами Coma и Partitur-Editor.

Языки программирования, которые мы использовали для написания веб-страницы, – HTML, CSS, JavaScript. Среда разработки – Visual Studio Code.

Ход работы

Наш проект можно условно поделить на 3 части: работа с метаданными записанных текстов, работа с самими текстами и визуализация на карте. Несмотря на то, что первые два пункта подразумевают под собой парсинг, они сильно отличались между собой, так как данные брались из файлов разных форматов. Поэтому эти части мы разделили между собой, чтобы потом объединить все в один файл, и заниматься самой визуализацией.

Метаданные, с которыми мы работали, были собраны в файлы формата .coma, который можно читать так же, как и любой файл формата .xml. Сначала нужно было разобраться со структурой и возможностями xml, поэтому сначала работа шла только с одним из корпусов.



Корпус метаданных долганского языка. Пример для текста

Первым этапом предполагалось извлечение основной информации о тексте и вывод полученных данных на форму, чтобы отследить правильность выполнения. Для решения этой задачи мы подключили библиотеку System.Xml, которая давала возможность работать непосредственно с файлами этого формата.

После этого нам нужно было полученные данные записать в файл формата .kml, который нам прислал заказчик и в котором уже были описаны некоторые метки поселков и городов, их координаты и некоторая информация к ним. Сначала мы искали библиотеку, которая позволяла бы легко работать с файлами этого формата, читать и записывать их. Библиотека SharpKml, которую мы в итоге нашли, была создана специально для этого, но в итоге мы решили не подключать ее, а пользоваться возможностями библиотеки System.Xml, которая подходит и для чтения файлов kml, чтобы создать такую же структуру для каждой точки с информацией о тексте.

Для каждой метки, которая должна была отобразиться на карте, необходимо было записать географические координаты (в KML они записываются в специальный тэг coordinates). У некоторых поселков координаты уже были записаны в метаданных, в таких случаях они извлекались вместе с другой информацией). Но большинство координат не имело, так что их нужно было получать с помощью поисково-информационных картографических сервисов вроде Google Maps или Яндекс.Карт по названию объекта. Сначала мы хотели использовать класс GeoCoordinate, но не нашли там возможности сделать то, что нам было нужно. Поэтому решили использовать библиотеку GMap.Net и

сервисом OpenStreetMap, так как именно там поиск по названию работал правильнее всего и находил больше необходимых объектов (на картах Google многих поселков не было вообще). Если после этого у поселка так и не было координат, мы искали их на других картах и меняли уже вручную в сохраненном kml-файле. Измененный файл открывали с помощью приложения Google Earth, которое позволяло сделать это быстро, не загружая карту на Google Maps для чтения.

Помимо меток о текстах нам также надо было отобразить на карте и другие метки, которые были бы связаны уже с рассказчиками. Всего таких меток было 3 вида: место рождения рассказчика (для каждого – одно), место проживания (тоже одно для одного человека) и все те места, которые рассказчик посетил в течение своей жизни (где учился, служил и тому подобное, отличное от места рождения и проживания).

Speaker: PoNA (Nikolaj Anisimovich Popov, Sex: male)	
Description (Speaker)	
1a Family name	Popov
1b Family name (RU)	Попов
2a Given name	Nikolaj
2b Given name (RU)	Николай
3a Patronymic	Anisimovich
3b Patronymic (RU)	Анисимович
4 Vulgo (Dolgan name)	...
5a Alternate names	...
5b Alternate names (RU)	...
4 Locations	
Basic biogr. data (Location)	
Description (Location)	
1a Place of birth	Volochanka
1b Place of birth (RU)	Волочанка
1c Place of birth (LngLat)	94.539288,70.976515
2 Region	Taymyr Peninsula
3 Country	Russia
4 Date of birth	1929.03.17.
5 Date of death	2009.03.12.
6a Former residences	Igarka, Leningrad/St. Petersburg
6b Former residences (RU)	Игарка, Ленинград/Санкт-Петербург
7a Domicile	Dudinka
7b Domicile (RU)	Дудинка
7c Domicile (LngLat)	86.158378,69.403488
8a Other information	...
8b Other information (RU)	...
Education (Location)	
Description (Location)	
1a Education	school
1b Education (RU)	школа
2a Higher education	pedagogic high school in Igarka, Herzen institute in Leningrad
2b Higher education (RU)	педагогический ВУЗ в Красноярске, Институт имени А.И. Герцена
3a Occupation	teacher, journalist, author

Пример для рассказчика

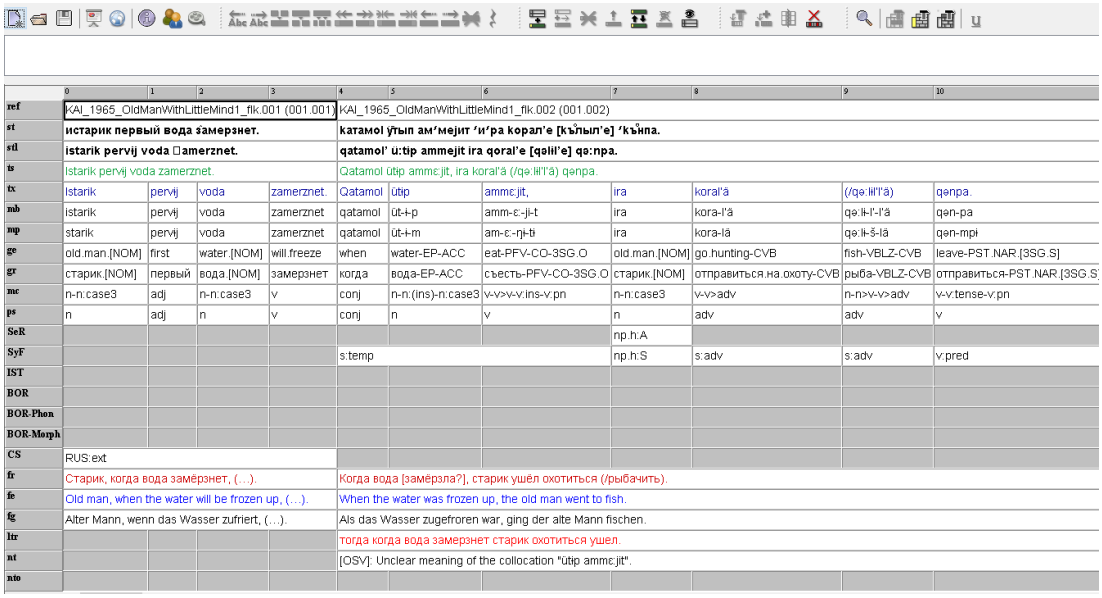
В метаданных вся эта информация была собрана для каждого рассказчика отдельно, и нам нужно было просто извлечь эту информацию вместе с другой (например, датой рождения и языками, на которых человек разговаривает).

Для каждого из этих мест мы создавали такие же метки, как и для текстов, но дополняли информацией о рассказчике. В некоторых случаях, когда посещенных мест было несколько и их было сложно извлечь, мы записывали их позже вручную.

После того, как мы протестировали программу на одном корпусе, мы добавили остальные два. Они были больше по размеру, поэтому программа очень долго извлекала и записывала информацию. Поэтому мы решили, что если мы больше не будем работать с

корпусами, то можно просто сохранить тот kml-файл, который мы получили на выходе и после, при создании путей-связей между точками, использовать его как входные данные.

После этого мы работали с уже текстами некоторых из рассказчиков на этих трех языках. Для каждого языка было дано по папке, внутри которых находились ещё папки с названиями текстов (которые строились из даты записи, ника рассказчика или рассказчиков, если это был диалог, и самого названия текста). В записанных рассказах встречались имена собственные. Задача была следующая — извлечь из текстов названия только географических объектов, для чего сначала нужно было извлечь все имена собственные, в том числе имена людей. Прежде всего нам нужно было разобраться с разметкой этих файлов (они были даны в формате .exb, открывались как файл .xml) и выявлением общих признаков у имен собственных, с чем нам помог заказчик.



	0	1	2	3	4	5	6	7	8	9	10
ref	KAI_1965_OldManWithLittleMind1_flk.001 (001.001)				KAI_1965_OldManWithLittleMind1_flk.002 (001.002)						
st	**истарик первый вода замерзнет.**				**qatamol ʉ̈tып ам′мejит ′и′ра коралʼе [къ̆лыпʼе] ′къ̆нпа.**						
stl	**istarik pervij voda □amerznet.**				**qatamol' ü:tip ammejit ira qoral'e [qəlii'e] qə:npa.**						
ts	Istarik pervij voda zamerznet.				Qatamol ütip ammɛ:jit, ira koral'ä (/qə:lil'l'ä) qənpa.						
tx	Istarik	pervij	voda	zamerznet.	Qatamol	ütip	ammɛ:jit,	ira	koral'ä	(/qə:lil'l'ä)	qənpa.
mb	istarik	pervij	voda	zamerznet	qatamol	üt-i-p	amm-ɛ:-j-i-t	ira	kora-l'ä	qə:li-l'-l'ä	qən-pa
mp	starik	pervij	voda	zamerznet	qatamol	üt-i-m	am-ɛ:-ŋ-ti	ira	kora-lä	qə:li-š-lä	qən-mpi
ge	old.man.[NOM]	first	water.[NOM]	will.freeze	when	water-EP-ACC	eat-PFV-CO-3SG.O	old.man.[NOM]	go.hunting-CVB	fish-VBLZ-CVB	leave-PST.NAR.[3SG:S]
gr	старик.[NOM]	первый	вода.[NOM]	замерзнет	когда	вода-EP-ACC	съесть-PFV-CO-3SG.O	старик.[NOM]	отправиться.на.охоту-CVB	рыба-VBLZ-CVB	отправиться-PST.NAR.[3SG:S]
mc	n-n:case3	adj	n-n:case3	v	conj	n-n:(ins)-n:case3	v-v>v-v:ins-v.pn	n-n:case3	v-v>adv	n-n>v-v>adv	v-v:tense-v.pn
ps	n	adj	n	v	conj	n	v	n	adv	adv	v
SeR								np.h:A			
SyF					s:temp			np.h:S	s:adv	s:adv	v:pred
IST											
BOR											
BOR-Phon											
BOR-Morph											
CS	RUS:ext										
fr	Старик, когда вода замёрзнет, (…).				Когда вода [замёрзла?], старик ушёл охотиться (/рыбачить).						
fe	Old man, when the water will be frozen up, (…).				When the water was frozen up, the old man went to fish.						
fg	Alter Mann, wenn das Wasser zufriert, (…).				Als das Wasser zugefroren war, ging der alte Mann fischen.						
ltr					тогда когда вода замерзнет старик охотиться ушел.						
nt					[OSV]: Unclear meaning of the collocation "ütip ammɛ:jit".						
nto											

Чтобы выделить названия из корпуса автоматически, мы нашли все слова, где в слое «ps», отвечающем за часть речи, указано, что это имя собственное (в текстах на долганском и камасинском языке это «propr», в текстах, написанных на селькупском – «nprop»). Затем из слоя «gr» («глоссирования» на русском языке), мы извлекали перевод этого фрагмента на русский язык (если были какие-то суффиксы или окончания, отделённые дефисами, мы их убирали). Для каждого найденного имени мы сохраняли строки из слоёв «ref» (название текста и номер предложения), «ts» (предложение в оригинале), «fr» (русский перевод предложения). Здесь возникла трудность с различием имен собственных и нарицательных. У них нет никаких отличительных свойств в данных файлах, поэтому следующим шагом был просмотр всех извлеченных имен собственных и удаление вручную нарицательных имён существительных и имен людей.

Так же, как и корпусами метаданных, сначала мы работали только с одной папкой языка, чтобы мы смогли разобраться и написать программу, записывающую все географические объекты. Потом мы подключили к программе три папки с текстами на долганском, камасинском и селькупском языках.

Координаты для этих географических объектов мы искали так же, как и для тех объектов, которые мы извлекали из метаданных: с помощью библиотеки GMap.Net и открытого сервиса OpenStreetMap. Из всей извлеченной информации и полученных координат мы сформировали отдельный файл .kml, который позже соединили kml-файл, в котором записаны места из метаданных. В дальнейшем мы использовали для работы этот полученный файл.

Также, чтобы наши метки не находились в одной точке на карте из-за одинаковых координат, так как это затрудняет работу с картой, не давая посмотреть все метки в этой точке, а только последнюю добавленную, нам нужно было немного сдвинуть их координаты, чтобы метки выглядели как скопление вокруг примерно одного и того же места. Изменяли мы широту и долготу точек прохождением через весь kml-файл и нахождением точек с одинаковыми координатами. При этом главным критерием смещения координат было и различие хотя бы одного признака в описании точки (в KML они находятся в специальном тэге description, куда заносится информация из тэга ExtendedData), чтобы избежать сравнение одной и той же точки. Различие могло быть или в дате записи (“recordDate”), или в названии (“titleEN”), или в имени человека, записывающего текст (“speakers”). После определения тех координат, которые надо изменить, редактировалась или широта, или долгота на 0,01. Изменение происходило только в тэге coordinates, по которому программа выставляет точки на карте, но в описании (тэг description) можно просмотреть исходные координаты.

После того, как у нас появился уже полностью готовый файл, который уже можно было отобразить на карте со всеми связями и всей информацией, мы перешли к визуализации. Хотя Google Earth и давало возможность показать все это в приложении, нам нужно было еще и реализовать что-то вроде поиска по странице (то есть, по меткам) и различных фильтров (по языкам, году рождения, дате записи и т.д.). Чтобы это сделать, необходимо создать веб-страницу, где вывести карту с помощью API какой-нибудь из уже существующих карт и нанести на нее созданный нами слой kml. Было две основные проблемы: во-первых, ключ для добавления API Google Maps, с которыми до этого мы работали, оказался платным, как и ключ Яндекс.Карт, поэтому нужно было найти что-то

открытое для общего пользования и при этом поддерживающее формат kml. Во-вторых, до этого мы никогда не работали с веб-страницами углубленно, поэтому решили создать веб-приложение на платформе ASP.NET, где могли продолжать работать на C#, почти не касаясь чего-то нового. Но после нескольких неудачных попыток создать на этой платформе хоть что-то, мы поняли, что, скорее всего, придется пробовать какой-то другой способ для реализации нашей задачи. После консультации пришли к следующему: будем использовать API 2GIS, к которому есть полное руководство, и писать веб-страницу на HTML, CSS и JavaScript для придания интерактивности.

Оказалось, что 2GIS, который мы хотели использовать, не имеет прямой поддержки файлов kml, то есть совершенно не подходит нам для работы. Можно было подключить внешний модуль Leaflet-plugin, но мы решили просто использовать API Leaflet, основанный на карте OpenStreetMap, которой мы пользовались и до этого.

Нам удалось подключить карту Leaflet к нашей будущей веб-странице. Следующим этапом нам нужно было сделать так, чтобы на базовую карту сверху накладывались слои, которые получались бы при парсинге kml-файлов. JavaScript не работает напрямую с KML, но для Leaflet есть специальный плагин, написанный специально для этих карт, который мы подключили для преобразования kml в xml и дальнейшей работы. С помощью этого плагина у нас получилось вывести общую карту со всеми метками и путями.

После этого мы стали работать над внешним видом страницы и добавлением специального сайдбара (sidebar — закрепленная или подвижная боковая панель, на которой размещаются разные навигационные данные сайта) с фильтрами и поисковой строкой. Хотелось сделать интуитивно-понятный для любого пользователя интерфейс, поэтому при оформлении мы опирались на самые популярные среди пользователей картографические сервисы: Google Maps, 2GIS, Яндекс.Карты. Все фильтры, которые были необходимы по задаче, мы разместили на подвижном сайдбаре, над которым разместили строку для поиска по странице. Сложнее всего было создать слайдер с двумя значениями: базового элемента для него в HTML нет, поэтому пришлось создавать второй ползунок и накладывать на первый, а потом писать отдельный алгоритм для того, чтобы не дать одному из «бегунков» зайти за другой.

После того, как все основное внешнее оформление было закончено, нам оставался последний этап работы: «активация» фильтров, чтобы при нажатии на них карта изменялась по заданному условию. Этот этап оказался одним из самых сложных, потому что под собой он подразумевал работу, в первую очередь, на языке JavaScript, с которым никто из нас до

этого не был знаком. Кроме этого, в Интернете было очень мало действительно полезной для нас информации о парсинге файлов XML с помощью JavaScript, в отличие от C#. Поэтому мы решили обойти эту проблему: разбили наш большой файл с картой для 3 языков на три файла (для каждого языка был свой файл) и подгружали каждый файл отдельным слоем для базовой карты при изменении какого-либо из флажков-чекбоксов.

Конечно, такой вариант не мог подойти ко всем другим фильтрам, ведь тогда нам нужно было бы создавать отдельные файлы для мужского и женского пола для каждого языка (то есть, у нас было бы уже 6 подгружающихся файлов) и для каждого года и возраста. Поэтому нам надо было искать способ создавать динамические слои, в которые данные подгружаются из общего файла только в том случае, если они удовлетворяют некоторому условию. Здесь уже был просто необходим парсинг, поэтому после неудачных попыток загрузить в программу файлы kml так же, как загружается обычный xml, мы обратились к коду плагина, который подключали в самом начале для работы с KML, чтобы посмотреть, как там решается задача парсинга. После изучения этого приложения нам удалось написать нечто похожее в своей программе, иногда со ссылками на некоторые функции плагина, чтобы не повторять их в своем коде. Мы снова соединили три отдельных файла в один большой и работали с ним.

Когда все фильтры были активными, нам необходимо было сделать так, чтобы все они были связаны друг с другом: то есть, если на карте у нас был выведен слой с долганским языком и мы хотели бы поставить промежуток лет записи от 2000 года до 2005, то фильтр на году работал только с отображенным на карте слоем. Чтобы решить эту задачу, необходимо было понять, как эти фильтры связаны друг с другом: какой зависит от какого и какой является основным. По сути, основным, конечно, был фильтр по языкам; без меток текстов не могли показываться метки рассказчиков или перерисовываться вся карта при движениях слайдеров. Поэтому для каждого флажка, отвечающего за конкретный язык, в функцию отображения мы внесли проверки на «отмеченный/неотмеченный» флажок фильтров по полу и проверки на изменения положения ползунков. Это, конечно, не самое оптимальное и правильное решение, но оно достаточно простое и полностью рабочее.

С поисковой строкой все было сложнее. Для того, чтобы сделать ее рабочей с функцией живого поиска (то есть, с выпадающим окном с подсказками при вводе) нужно создавать что-то вроде базы данных, которая тоже будет изменяться по фильтрам. Кроме того, для работы с такими функциями нужно дополнительно изучать AJAX.

Еще одна сложность, с которой мы столкнулись, заключалась в том, что пути, которые мы проводили из одной точки в другую и которые были тоже записаны в файле kml, должны были динамически перерисовываться, если точка, к которой они проведены, удалялась с карты. Здесь тоже требовался парсинг файла, причем гораздо более сложный и затратный. Мы решили пока этим не заниматься, а чтобы пути не оставались просто висящими на карте без каких-либо точек, сделали что-то вроде движка «включено/выключено».

Распределение обязанностей

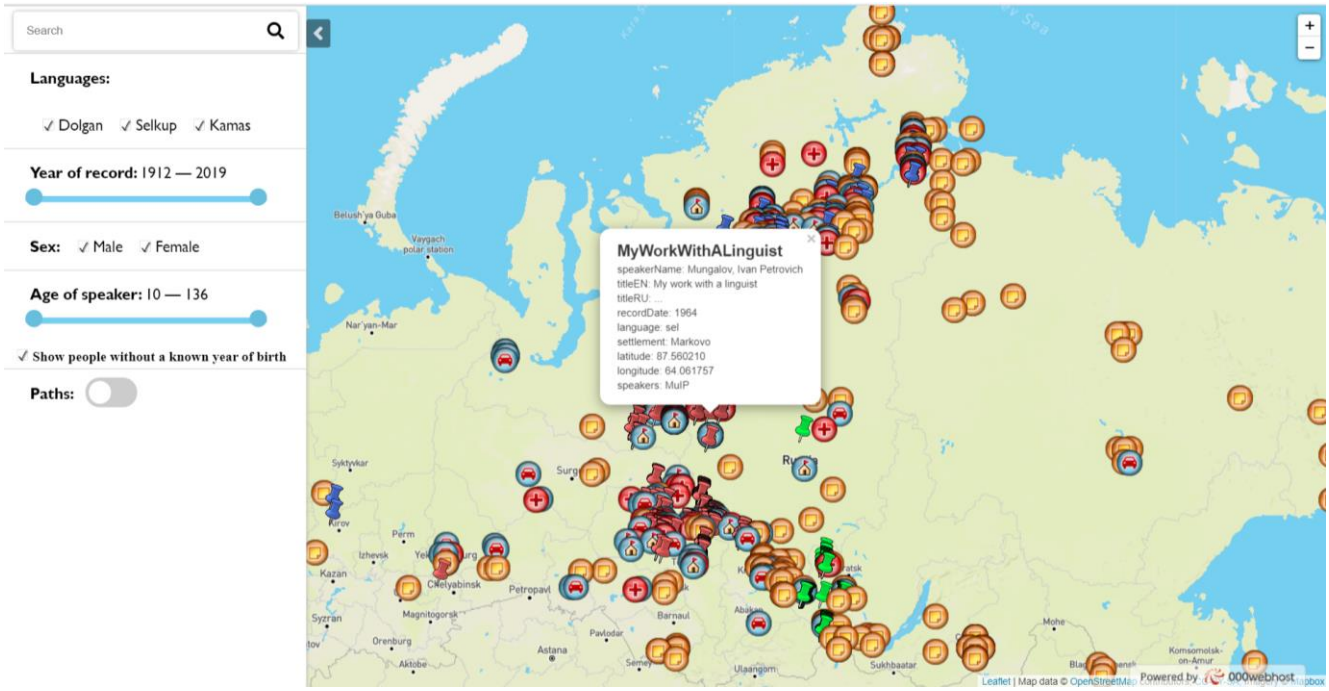
Леоненкова Виктория:

- Работала с файлами .exb и извлекала из них всех географических объектов и информации о них в файл .kml
- Написала программу для сдвига координат, чтобы не было наложения
- Занималась проведением путей между метками

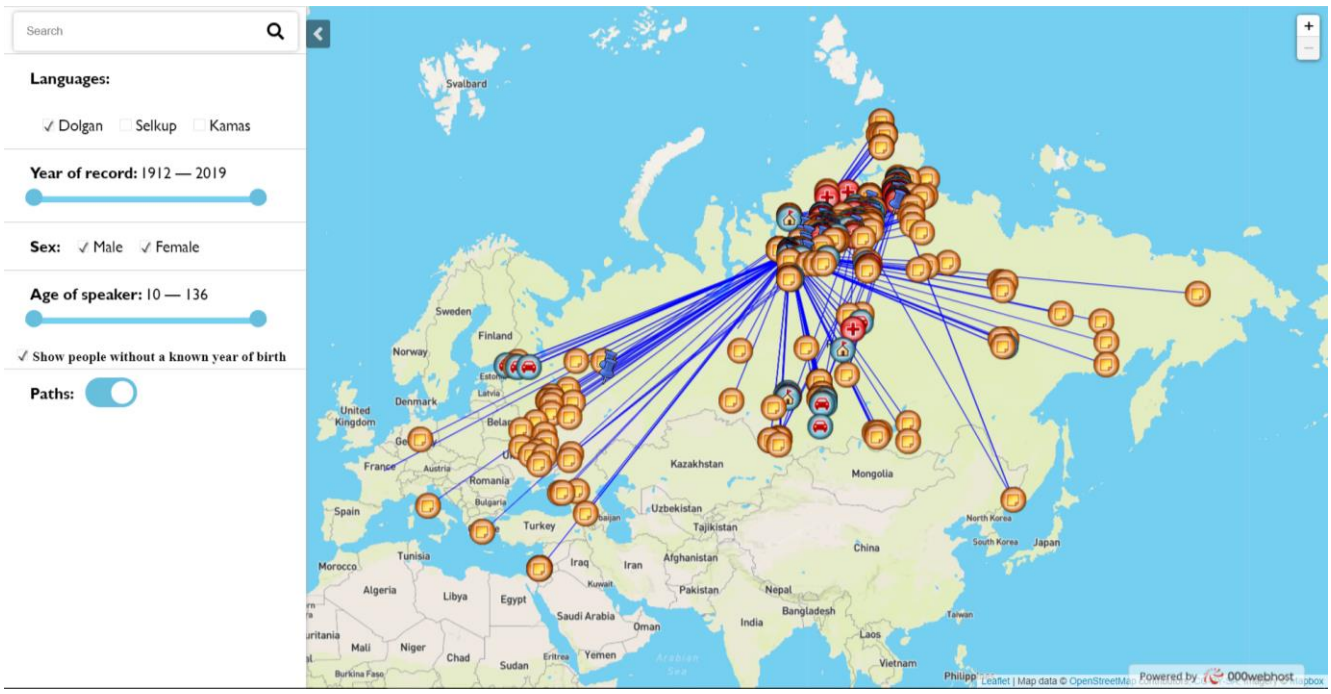
Потапова Дарья:

- Работала с метаданными и извлекала оттуда всю информацию о текстах и местах рождения, проживания и посещения рассказчиков в файл .kml
- Создала страницу сайта для визуализации и подключила к нему полученный kml-файл
- Написала фильтры, взаимодействующие с картой и подключенным файлом

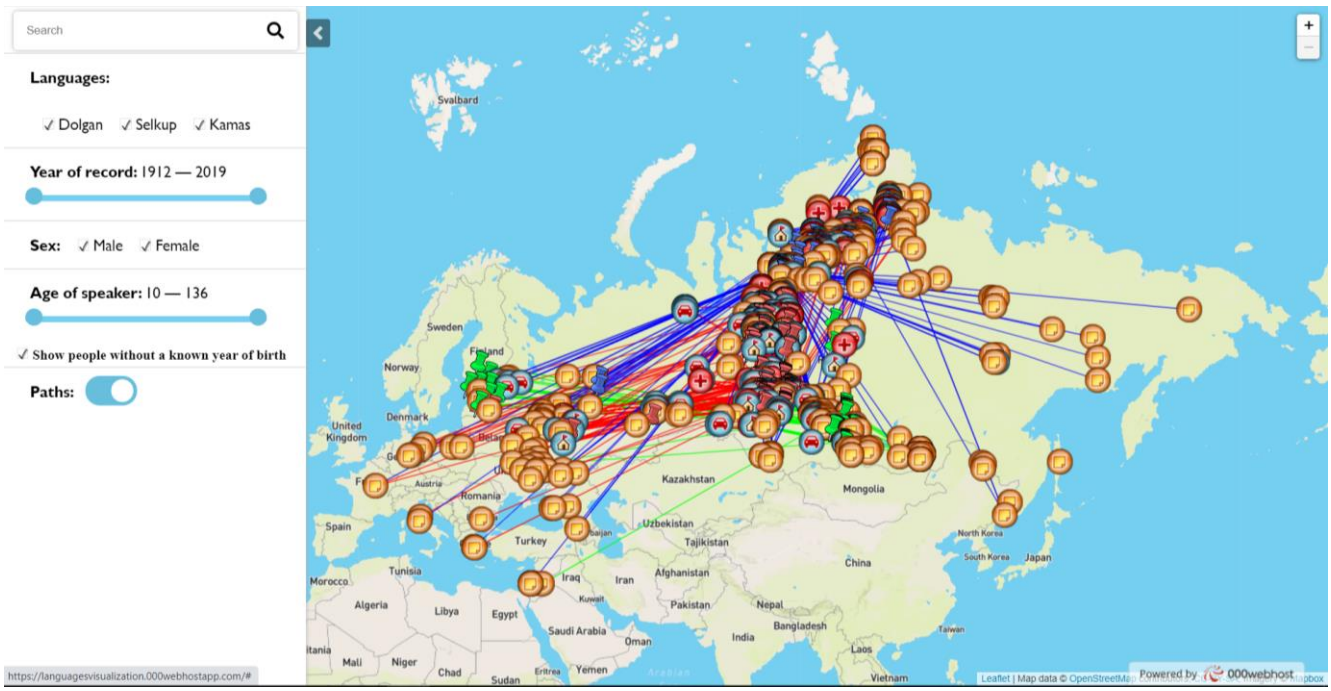
Примеры использования программы



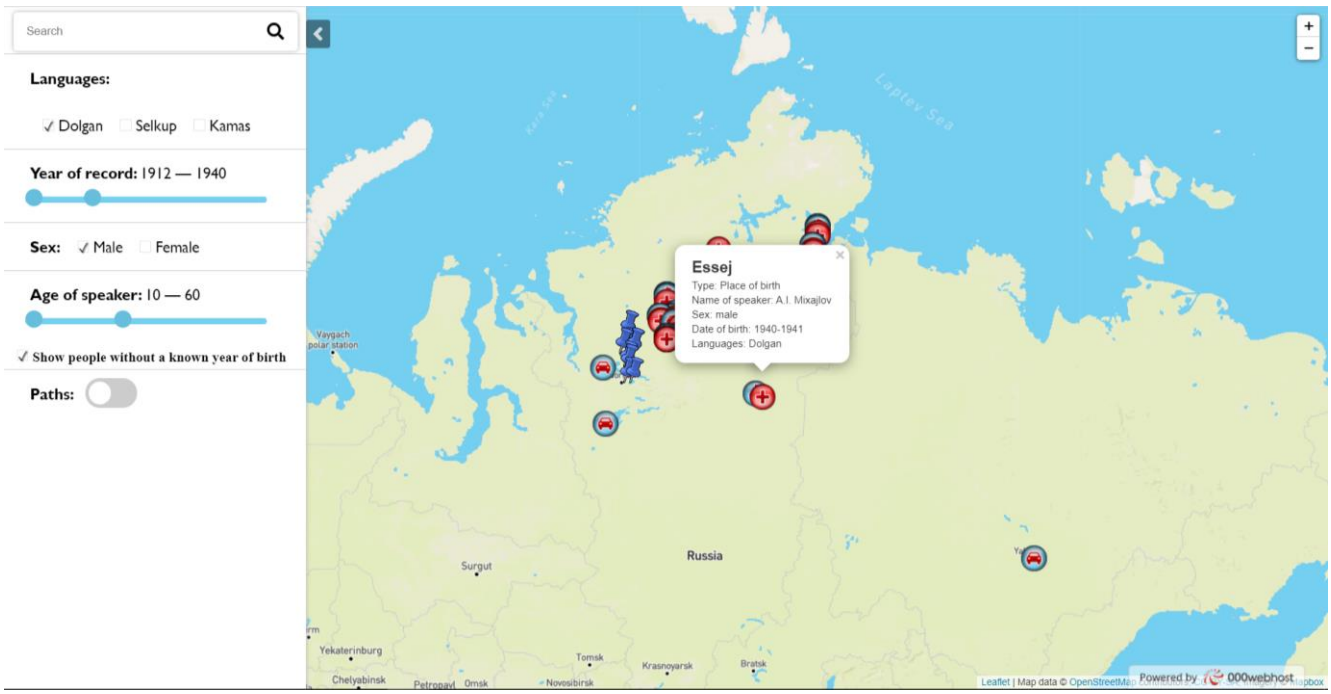
Меню каждой метки текста. Пути выключены.



Все пути долганского языка. Из-за большого количества меток получается нагромождение путей, которые мы попытались избежать, добавив фильтр для отображение выбранного языка.



Все пути и языки нанесены на карту. Включены все фильтры.



Точки долганских текстов, записанных в промежутке от 1912 года до 1940. Метки рассказчиков мужского пола в возрасте до 60 лет. Пути выключены.

Результат работы

В результате нашей работы у нас получилось выполнить почти все требования, который ставил нам заказчик: у нас есть веб-страница с картой и понятным интерфейсом, возможность вывести или убрать те или иные метки с помощью фильтров (выбор языка, года записи текста, пола или рассказчика).

В процессе работы мы изучили довольно много нового и полезного для себя материала: научились работать с файлами разных форматов: .coma, .exb, .xml и, конечно, .kml. Мы узнали о том, как можно парсить xml-файлы, как получать координаты с помощью внешних серверов; познакомились с новым для нас языком программирования JavaScript и научились оформлять простые веб-страницы, работать с картами, метками, слоями и путями.

Перспективы дальнейшей разработки

Конечно, у нашей работы есть куда развиваться. Во-первых, код для извлечения информации метаданных, скорее всего, можно сделать более универсальным для любого нового корпуса (сейчас нам все-таки приходилось немного его изменять для каждого корпуса, потому что они собраны не по одному принципу, например, способ записи даты рождения мог быть разных форматов: «дд.мм.гггг», «гггг.мм.дд» или «гггг»). Во-вторых, можно совершенствовать саму веб-страницу: добавить новые фильтры, сделать строку поиска активной, сделать для слайдеров окошки для ввода значений и подстраивание движка по этому значению. В-третьих, нам пока не удалось решить проблему отображения путей только при нажатии на точку, и это, конечно, тоже уйдет на дальнейшую разработку, так как эта проблема нуждается в решении для облегчения восприятия материала, отображенного на карте. В-четвертых, код программы, написанный на JavaScript, скорее всего, можно сильно упростить, если углубленно заняться этим языком и изучить его лучше. В-пятых, если в исходные добавить метки, помогающие отличить имена, названия ресторанов, спортивных центров, гостиниц и т. п. от географических объектов, то можно автоматизировать процесс отбора географических названий из всех имен собственных.

Список источников и литературы

Мы пользовались следующими источниками для решения нашей задачи:

1. StackOverFlow [Электронный ресурс] : многопредмет. форум — Форум по программированию — Режим доступа: https://stackoverflow.com/
2. Cyberforum [Электронный ресурс] : многопредмет. форум – Форум программистов и сисадминов — Режим доступа : http://www.cyberforum.ru/
3. Язык разметки Keyhole [Электронный ресурс] : официальная документация — Справочник по KML Reference — Режим доступа: https://developers.google.com/kml?hl=ru/
4. Leaflet [Электронный ресурс] : официальная документация — Open-source JavaScript library for mobile-friendly interactive maps — Режим доступа: https://leafletjs.com/
5. Learn JavaScript [Электронный ресурс] : современный учебник JavaScript — Режим доступа: https://learn.javascript.ru/
6. GitHub [Электронный ресурс] : ресурс для разработчиков — Режим доступа: https://github.com/Leaflet/Leaflet
7. OpenStreetMap [Электронный ресурс] : веб-картографический проект — Режим доступа: https://www.openstreetmap.org/#map=5/58.136/99.448

Приложения

1) Извлечение информации из метаданных и запись KML-файла

```
    class Program
    {
        public static string getNodeValue(XmlDocument xmldoc, string id, string key)
        {
            return
xmldoc.SelectSingleNode($"Corpus/CorpusData/Communication[@Id='{id}']/Description/Key[@Name='{
key}']").InnerText;
        }

        public static string getSpeakerNodeValue(XmlDocument xmldoc, string id, string key)
        {
            if (id != "SID4F3B7EEF-2A80-C592-81DB-751C432655F8")
            {
                return
xmldoc.SelectSingleNode($"Corpus/CorpusData/Speaker[@Id='{id}']/Location[@Type='{"Basic biogr.
data"}']/Description/Key[@Name='{key}']").InnerText;
            }
            return "...";
        }

        public static string getLocation(XmlDocument xmldoc, string id, string key)
        {
            if (id != "CID6D7C8E7B-C13E-7FA7-2B14-07DD55D61E68")
            {
                return
xmldoc.SelectSingleNode($"Corpus/CorpusData/Communication[@Id='{id}']/Location/Description/Key
[@Name='{key}']").InnerText;
            }
            else
            {
                return
xmldoc.SelectSingleNode($"Corpus/CorpusData/Communication[@Id='{id}']/Location/City").InnerTex
t;
            }
        }

        public static string getLanguage(XmlDocument xmldoc, string id)
        {
            return
xmldoc.SelectSingleNode($"Corpus/CorpusData/Communication[@Id='{id}']/Language/LanguageCode").
InnerText;
        }

        public static string getX(XmlDocument xmldoc, string id, string sttl)
        {
            if ((sttl == "Dudinka" || sttl == "Volochanka") && id != "CID6D7C8E7B-C13E-7FA7-
2B14-07DD55D61E68")
            {
                string line =
xmldoc.SelectSingleNode($"Corpus/CorpusData/Communication[@Id='{id}']/Location/Description/Key
[@Name='Settlement (LngLat)']").InnerText;
                return line.Substring(0, line.IndexOf(","));
            }
            else
            {
                if (id == "CID6D7C8E7B-C13E-7FA7-2B14-07DD55D61E68")
                {
                    return "86.158378";
                }
```

```
                return "0";
            }
        }

        public static string getY(XmlDocument xmldoc, string id, string sttl)
        {
            if ((sttl == "Dudinka" || sttl == "Volochanka") && id != "CID6D7C8E7B-C13E-7FA7-
2B14-07DD55D61E68")
            {
                string line =
xmldoc.SelectSingleNode($"Corpus/CorpusData/Communication[@Id='{id}']/Location/Description/Key
[@Name='Settlement (LngLat)']").InnerText;
                return line.Substring(line.IndexOf(",") + 1, line.Length - line.IndexOf(",") -
1);
            }
            else
            {
                if (id == "CID6D7C8E7B-C13E-7FA7-2B14-07DD55D61E68")
                {
                    return "69.403488";
                }
                return "0";
            }
        }

        static string getMapValue(XmlDocument xmldoc)
        {
            if (xmldoc["kml"]["Document"]["Placemark"]["name"].InnerText == null)
            {
                return "";
            }
            else
            {
                return xmldoc.SelectSingleNode($"kml/Document/Placemark/name").InnerText;
            }
        }

        public static string getLanguage(string l)
        {
            switch (l)
            {
                case "dlg":
                    {
                        return "dlg.png";
                    }

                case "xas":
                    {
                        return "xas.png";
                    }

                case "sel":
                    {
                        return "sel.png";
                    }
            }
            return null;
        }

        public static void Speaker(XmlDocument kmlMap, List<SpeakerInfo> spInfos, string key)
        {
            foreach (SpeakerInfo sp in spInfos)
            {
                XmlElement placeNode = kmlMap.CreateElement("Placemark");
```

```
                kmlMap["kml"]["Document"].AppendChild(placeNode);

                XmlElement birthPl = kmlMap.CreateElement("name");
                XmlText birthPlName = kmlMap.CreateTextNode(sp.placeOfBirth);
                placeNode.AppendChild(birthPl).AppendChild(birthPlName);

                XmlElement highlightStyle = kmlMap.CreateElement("Style");
                highlightStyle.SetAttribute("id", "highlightPlacemark");
                XmlElement IconStyleH = kmlMap.CreateElement("IconStyle");
                XmlElement IconH = kmlMap.CreateElement("Icon");
                XmlElement link = kmlMap.CreateElement("href");

placeNode.AppendChild(highlightStyle).AppendChild(IconStyleH).AppendChild(IconH).AppendChild(l
ink);
                XmlText linkText = kmlMap.CreateTextNode("");
                if (key == "Date of birth") { linkText = kmlMap.CreateTextNode("birth.png"); }
                if (key == "Domicile") { linkText = kmlMap.CreateTextNode("domicile.png"); }
                link.AppendChild(linkText);

                XmlElement descNode = kmlMap.CreateElement("description");
                placeNode.AppendChild(descNode);

                XmlElement pointNode = kmlMap.CreateElement("Point");
                placeNode.AppendChild(pointNode);

                XmlElement coordNode = kmlMap.CreateElement("coordinates");
                XmlText coordText = kmlMap.CreateTextNode(String.Format("{0},{1}",
sp.latitude.Replace(",", "."), sp.longitude.Replace(",", ".")));
                pointNode.AppendChild(coordNode);
                coordNode.AppendChild(coordText);

                XmlElement exData = kmlMap.CreateElement("ExtendedData");
                placeNode.AppendChild(exData);

                XmlElement data, value; XmlText valueText;
                data = kmlMap.CreateElement("Data");
                value = kmlMap.CreateElement("value");
                valueText = kmlMap.CreateTextNode("");
                exData.AppendChild(data).AppendChild(value).AppendChild(valueText);

                data = kmlMap.CreateElement("Data");
                data.SetAttribute("name", "Type");
                value = kmlMap.CreateElement("value");
                valueText = kmlMap.CreateTextNode(key);
                exData.AppendChild(value).AppendChild(valueText);

                data = kmlMap.CreateElement("Data");
                data.SetAttribute("name", "Name of speaker");
                value = kmlMap.CreateElement("value");
                valueText = kmlMap.CreateTextNode(sp.speakerName);
                exData.AppendChild(data).AppendChild(value).AppendChild(valueText);

                data = kmlMap.CreateElement("Data");
                data.SetAttribute("name", "Sex");
                value = kmlMap.CreateElement("value");
                valueText = kmlMap.CreateTextNode(sp.sex);
                exData.AppendChild(data).AppendChild(value).AppendChild(valueText);

                data = kmlMap.CreateElement("Data");
                data.SetAttribute("name", "Date of birth");
                value = kmlMap.CreateElement("value");
                valueText = kmlMap.CreateTextNode(sp.dateOfBirth);
                exData.AppendChild(data).AppendChild(value).AppendChild(valueText);
            }
```

```
        }

        static void Main(string[] args)
        {
            string titleEN = "", titleRU = "", date = "", language = "", settlement = "",
latitude = "", longitude = "", speakers = "";
            string sigle = "", speakerName = "", sex = "", birthDate = "", birthPlace = "",
domicile = "";
            List<Info> infos = new List<Info>();
            List<SpeakerInfo> spInfos = new List<SpeakerInfo>();
            LoadInfo();
            SpeakerInfo();
            Map();

            void LoadInfo()
            {
                List<XmlDocument> doc = new List<XmlDocument>();

                XmlDocument d1 = new XmlDocument();
                d1.Load("dolgan-v10.coma");
                doc.Add(d1);
                d1 = new XmlDocument();
                d1.Load("selkup.coma");
                doc.Add(d1);
                d1 = new XmlDocument();
                d1.Load("kamas.coma");
                doc.Add(d1);

                foreach (XmlDocument dc in doc)
                {
                    XmlNodeList nodeList = dc.GetElementsByTagName("Communication");

                    foreach (XmlNode node in nodeList)
                    {
                        string id = node.Attributes[0].Value;
                        string person = "";
                        if (node["Setting"]["Person"] != null)
                        {
                            person = node["Setting"]["Person"].InnerText;
                        }

                        titleEN = getNodeValue(dc, id, "0a Title");
                        titleRU = getNodeValue(dc, id, "0b Title (RU)");
                        date = getNodeValue(dc, id, "2b Date of recording");
                        speakers = getNodeValue(dc, id, "4 Speakers");
                        language = getLanguage(dc, id);
                        settlement = getLocation(dc, id, "Settlement");
                        latitude = getX(dc, id, settlement).Replace(",", "."); //широта
                        longitude = getY(dc, id, settlement).Replace(",", "."); //долгота

                        if (latitude == "0" && longitude == "0")
                        {
                            string helpset = "";
                            if (settlement == "Xatanga")
                            {
                                helpset = "Деревня Хатанга";
                            }
                            else
                            {
                                if (settlement == "Zhdanixa") helpset = "Посёлок Жданиха";
                                if (settlement == "Kresty`") helpset = "Фактория Кресты";
                                else
                                {
                                    helpset = settlement;
```

```
                            }
                        }

                        PointLatLng? pointLatLng =
GMapProviders.OpenStreetMap.GetPoint(helpset, out GeoCoderStatusCode status);
                        if (status == GeoCoderStatusCode.OK)
                        {
                            longitude = pointLatLng.Value.Lat.ToString().Replace(",",
".");
                            latitude = pointLatLng.Value.Lng.ToString().Replace(",", ".");
                        }
                        else
                        {
                            if (settlement == "Stanok Kry`s at river Pyasina")
                            {
                                longitude = "70.575870";
                                latitude = "89.066104";
                            }
                            else
                            {
                                if (settlement == "Xeta")
                                {
                                    longitude = "71.5634237";
                                    latitude = "99.6524644";
                                }
                            }
                            //MessageBox.Show("Failed");
                        }
                    }
                    infos.Add(new Info(speakerName, titleEN, titleRU, date, language,
settlement, latitude, longitude, speakers));
                }
            }
        }

        void SpeakerInfo()
        {
            List<XmlDocument> doc = new List<XmlDocument>();

            XmlDocument d1 = new XmlDocument();
            d1.Load("dolgan-v10.coma");
            doc.Add(d1);

            SpeakerInfo spInfo = new SpeakerInfo();
            d1 = new XmlDocument();
            d1.Load("selkup.coma");
            doc.Add(d1);
            d1 = new XmlDocument();
            d1.Load("kamas.coma");
            doc.Add(d1);

            foreach (XmlDocument dc in doc)
            {
                XmlNodeList speakerList = dc.GetElementsByTagName("Speaker");

                foreach (XmlNode node in speakerList)
                {
                    string id = node.Attributes[0].Value;
                    sigle = node["Sigle"].InnerText;
                    speakerName = node["Pseudo"].InnerText;
                    sex = node["Sex"].InnerText;


                    birthDate = getSpeakerNodeValue(dc, id, "4 Date of birth");
```

```
                        birthPlace = getSpeakerNodeValue(dc, id, "1a Place of birth");
                        domicile = getSpeakerNodeValue(dc, id, "7a Domicile");

                        PointLatLng? pointLatLng =
GMapProviders.OpenStreetMap.GetPoint(birthPlace, out GeoCoderStatusCode status);
                        if (status == GeoCoderStatusCode.OK)
                        {
                            longitude = pointLatLng.Value.Lat.ToString().Replace(",", ".");
                            latitude = pointLatLng.Value.Lng.ToString().Replace(",", ".");
                        }
                        spInfos.Add(new SpeakerInfo(speakerName, sex, birthDate, birthPlace,
domicile, latitude, longitude));
                    }

                }
            }

            void Map()
            {
                XmlDocument kmlMap = new XmlDocument();
                kmlMap.Load("INEL_LangsRecolored.kml");
                _ = kmlMap.GetElementsByTagName("Placemark");

                 foreach (Info info in infos)
                 {
                     XmlElement placeNode = kmlMap.CreateElement("Placemark");
                     kmlMap["kml"]["Document"].AppendChild(placeNode);

                     XmlElement nameNode = kmlMap.CreateElement("name");
                     XmlText nameText = kmlMap.CreateTextNode(info.titleEN);
                     placeNode.AppendChild(nameNode);
                     nameNode.AppendChild(nameText);
                     XmlElement highlightStyle = kmlMap.CreateElement("Style");
                     highlightStyle.SetAttribute("id", "highlightPlacemark");
                     XmlElement IconStyleH = kmlMap.CreateElement("IconStyle");
                     XmlElement IconH = kmlMap.CreateElement("Icon");
                     XmlElement link = kmlMap.CreateElement("href");

placeNode.AppendChild(highlightStyle).AppendChild(IconStyleH).AppendChild(IconH).AppendChild(l
ink);
                     XmlText linkText = kmlMap.CreateTextNode(getLanguage(info.language));

                     link.AppendChild(linkText);

                     XmlElement descNode = kmlMap.CreateElement("description");
                     placeNode.AppendChild(descNode);

                     XmlElement pointNode = kmlMap.CreateElement("Point");
                     placeNode.AppendChild(pointNode);

                     XmlElement coordNode = kmlMap.CreateElement("coordinates");
                     XmlText coordText = kmlMap.CreateTextNode(String.Format("{0},{1}",
info.latitude.Replace(",", "."), info.longitude.Replace(",", ".")));
                     pointNode.AppendChild(coordNode);
                      coordNode.AppendChild(coordText);

                     XmlElement exData = kmlMap.CreateElement("ExtendedData");
                     placeNode.AppendChild(exData);

                     foreach (var dataNames in typeof(Info).GetProperties())
                     {
                         XmlElement data = kmlMap.CreateElement("Data");
                         data.SetAttribute("name", dataNames.Name);
                         XmlElement value = kmlMap.CreateElement("value");
```

```
                        XmlText valueText =
kmlMap.CreateTextNode(dataNames.GetValue(info).ToString());
                        exData.AppendChild(data).AppendChild(value).AppendChild(valueText);
                    }
                }
                Speaker(kmlMap, spInfos, "Date of birth");
                kmlMap.Save("map.kml");

                Speaker(kmlMap, spInfos, "Domicile");
                kmlMap.Save("map.kml");
                Process.Start("map.kml");
            }
        }
    }
}
```

Классы:

```
class Info
    {
       // public string id { get; private set; }
        public string spikerName { get; set; }
        public string titleEN { get; set; }
        public string titleRU { get; set; }
        public string recordDate { get; set; }
        public string language { get; set; }
        public string settlement { get; set; }
        public string latitude { get; set; }
        public string longitude { get; set; }
        public string speakers { get; set; }

        public Info() { }
        public Info(string name, string titleEN, string titleRU, string date, string language,
string settlement, string latitude, string longitude, string speakers)
        {
            //this.id = id;
            spikerName = name;
            this.titleEN = titleEN;
            this.titleRU = titleRU;
            recordDate = date;
            this.language = language;
            this.settlement = settlement;
            this.longitude = longitude;
            this.latitude = latitude;
            this.speakers = speakers;
        }

        public override string ToString()
        {
            return titleEN;
        }
    }

class SpeakerInfo
    {
        public string speakerName { get; set; }
        public string sex { get; set; }
        public string dateOfBirth { get; set; }
        public string placeOfBirth { get; set; }
        public string domicile { get; set; }
        public string latitude { get; set; }
        public string longitude { get; set; }
```

```
        public SpeakerInfo() { }

        public SpeakerInfo(string speaker, string sex, string date, string birthPlace, string
domicile, string latitude, string longitude)
        {
            speakerName = speaker;
            this.sex = sex;
            dateOfBirth = date;
            placeOfBirth = birthPlace;
            this.domicile = domicile;
            this.longitude = longitude;
            this.latitude = latitude;
        }
    }
```

2) Извлечение информации из текстов и запись KML-файла

```
    class Program
    {
        public static string getText(string txt)
        {
            string ans = ""; bool flag = true; int i = 0;
            int index = txt.IndexOf("_", 0), num = 1;
            while (index > -1)
            {
                string res = "1" + txt[index + 1];
                if (num == 2 && !int.TryParse(res, out int result1)) { ans = txt.Remove(0,
index + 1); i = index + 1; }
                if (num == 3 && ans != "") { ans = ans.Remove(index - i, ans.Length - (index -
i)); flag = false; }
                if (num == 3 && flag) { ans = txt.Remove(0, index + 1); i = index + 1; }
                if (num == 4 && flag) ans = ans.Remove(index - i, ans.Length - (index - i));
                index = txt.IndexOf("_", index + 1);
                num++;
            }
            return ans;
        }

        public static string getDate(string txt)
        {
            string ans = ""; bool flag = true;
            int index = txt.IndexOf("_", 0), num = 1;
            while (index > -1)
            {
                string res = "1" + txt[index + 1];
                if (num == 1 && int.TryParse(res, out int result1)) { ans = txt.Remove(0,
index + 1); ans = ans.Remove(4, ans.Length - 4); flag = false; }
                if (num == 2 && flag) { ans = txt.Remove(0, index + 1); ans = ans.Remove(4,
ans.Length - 4); }
                index = txt.IndexOf("_", index + 1);
                num++;
            }
            return ans;
        }

        public static void AllNames(List<string> d, int dir, List<Names> names, string s)
        {
            XmlDocument doc = new XmlDocument();
            int begin = 0; bool samename = true; string path = "", language="";
            switch (dir)
            {
                case 0:
                    {
```

```
                        path = "SelkupCorpus-v01\\" + d[0] + "\\" + d[1] + "\\" + d[2] + "\\"
+ d[3];
                        language="selkup";
                        doc.Load(path); break;
                    }
                case 1:
                    {
                        path = "dolgan-noAudio\\" + d[0] + "\\" + d[1] + "\\" + d[2];
                        language = "dolgan";
                        doc.Load(path); break;
                    }
                case 2:
                    {
                        path = "kamas_noAudio\\" + d[0] + "\\" + d[1] + "\\" + d[2];
                        language = "kamas";
                        doc.Load(path); break;
                    }
            }
            XmlNodeList xnodeList = doc.GetElementsByTagName("tier");
            XmlNodeList nodeList = doc.GetElementsByTagName("event");
            int i = 0; List<string> speaker = new List<string>();
            foreach (XmlNode xnode in xnodeList)
            {
                foreach (XmlNode childnode in xnode.ChildNodes)
                {
                    if (xnode.Attributes["category"].Value.Contains("ps") &&
childnode.Name.Contains("event") && childnode.InnerText == s)
                    {
                        names.Add(new Names());
                        begin++;
                        names[names.Count - 1].start =
childnode.Attributes["start"].InnerText;
                        names[names.Count - 1].end = childnode.Attributes["end"].InnerText;
                        names[names.Count - 1].speaker = xnode.Attributes["speaker"].Value;
                        names[names.Count - 1].language = language;
                        if (speaker.Count == 0)
speaker.Add(xnode.Attributes["speaker"].Value);
                        if (xnode.Attributes["speaker"].Value != speaker[speaker.Count - 1])
speaker.Add(xnode.Attributes["speaker"].Value);
                    }
                }
            }
            i = names.Count - begin;
            if (begin != 0)
            {
                foreach (string sp in speaker)
                {
                    foreach (XmlNode xnode in xnodeList)
                    {
                        foreach (XmlNode childnode in xnode.ChildNodes)
                        {
                            if (xnode.Attributes["category"].Value == "gr" &&
childnode.Name.Contains("event") && childnode.Attributes["start"].InnerText == names[i].start
&& names[i].end == childnode.Attributes["end"].InnerText && names[i].speaker == sp)
                            {
                                string name = childnode.InnerText; int indexdot = 0, indexdash
= 0;
                                int index = 0; List<int> ind = new List<int>();
                                if (name.Contains("."))
                                {
                                    while ((index = name.IndexOf(".", index)) != -1)
                                    {
                                        ind.Add(index);
                                        index++;
```

```
                                    }
                                    index = 0;
                                    foreach (int tt in ind)
                                    {
                                        if (name[tt + 1] >= 'A' && name[tt + 1] <= 'Z' ||
name[tt + 1] == '[')
                                        {
                                            if (index == 0 || tt < index) index = tt;
                                        }
                                    }
                                    names[i].name = name.Remove(name.IndexOf("."));
                                    indexdot = name.IndexOf(".");
                                }
                                string res = "1" + name[name.IndexOf("-") + 1];
                                if (name.Contains("-") && (name[name.IndexOf("-") + 1] >= 'A'
&& name[name.IndexOf("-") + 1] <= 'Z') || int.TryParse(res, out int result1))
                                {
                                    names[i].name = name.Remove(name.IndexOf("-"));
                                    indexdash = name.IndexOf("-");
                                }
                                if (indexdot == 0 && indexdash == 0)
                                    names[i].name = name;
                                if (indexdot != 0 && indexdash != 0)
                                {
                                    if (indexdot < indexdash) names[i].name =
name.Remove(name.IndexOf("."));
                                    if (indexdash < indexdot) names[i].name =
name.Remove(name.IndexOf("-"));
                                }
                                if (names[i].name.Contains("?")) names[i].name =
name.Remove(name.IndexOf("?"));
                                for (int var = names.Count - begin; var < names.Count - 1;
var++)
                                {
                                    if (names[i].name == names[var].name && i != var &&
names[i].speaker == names[var].speaker) samename = false;
                                }
                                if (!samename)
                                {
                                    names.RemoveAt(i);
                                    i--; begin--;
                                }
                                samename = true;
                                if (i + 1 < names.Count || names.Count == 0) i++;
                            }
                        }
                    }
                }
                i = names.Count - begin;
                for (; i < names.Count; i++)
                {
                    foreach (string sp in speaker)
                    {
                        foreach (XmlNode xnode in xnodeList)
                        {
                            foreach (XmlNode childnode in xnode.ChildNodes)
                            {
                                string start =
childnode.Attributes["start"].InnerText.TrimStart('T');
                                string end =
childnode.Attributes["end"].InnerText.TrimStart('T');
                                if (xnode.Attributes["category"].Value == "ref" &&
childnode.Name.Contains("event") && int.Parse(names[i].start.TrimStart('T')) >=
```

```
int.Parse(start) && int.Parse(names[i].start.TrimStart('T')) + 1 <= int.Parse(end) &&
names[i].speaker == sp)
                                {
                                    string refname = childnode.InnerText;
                                    int index = 0; List<int> ind = new List<int>();
                                    while ((index = refname.IndexOf(".", index)) != -1)
                                    {
                                        ind.Add(index);
                                        index++;
                                    }
                                    index = 0;
                                    foreach (int tt in ind)
                                    {
                                        string res = "1" + refname[tt + 1];
                                        if (int.TryParse(res, out int result11))
                                        {
                                            if (index == 0 || tt < index) index = tt;
                                        }
                                    }
                                    names[i].refname = refname.Remove(index);
                                    string refnum = childnode.InnerText.Substring(index + 1);
                                    if (refnum.Contains(" ")) names[i].refnum =
int.Parse(refnum.Remove(refnum.IndexOf(" ")));
                                }
                                if (xnode.Attributes["category"].Value == "ts" &&
childnode.Name.Contains("event") && int.Parse(names[i].start.TrimStart('T')) >=
int.Parse(start) && int.Parse(names[i].start.TrimStart('T')) + 1 <= int.Parse(end) &&
names[i].speaker == sp)
                                {
                                    names[i].ts = childnode.InnerText;
                                }
                                if (xnode.Attributes["category"].Value == "fr" &&
childnode.Name.Contains("event") && int.Parse(names[i].start.TrimStart('T')) >=
int.Parse(start) && int.Parse(names[i].start.TrimStart('T')) + 1 <= int.Parse(end) &&
names[i].speaker == sp)
                                {
                                    names[i].fr = childnode.InnerText;
                                }
                                if (xnode.Attributes["category"].Value == "ltr" && names[i].fr
== null && childnode.Name.Contains("event") && int.Parse(names[i].start.TrimStart('T')) >=
int.Parse(start) && int.Parse(names[i].start.TrimStart('T')) + 1 <= int.Parse(end) &&
names[i].speaker == sp)
                                {
                                    names[i].fr = childnode.InnerText;
                                }
                            }
                        }
                    }
                }
            }
            begin = 0;
        }
        static void Main(string[] args)
        {
            List<Names> names = new List<Names>();
            DirectoryInfo directoryInfo = new DirectoryInfo("SelkupCorpus-v01");
            DirectoryInfo directoryInfo1 = new DirectoryInfo("dolgan-noAudio");
            DirectoryInfo directoryInfo2 = new DirectoryInfo("kamas_noAudio");
            List<DirectoryInfo> directories = new List<DirectoryInfo>();
            directories.Add(directoryInfo);
            directories.Add(directoryInfo1);
            directories.Add(directoryInfo2);
            for (int dir = 0; dir < directories.Count; dir++)
            {
```

```
                foreach (var file in directories[dir].GetDirectories())
                {
                    if (dir == 0)
                    {
                        foreach (var file1 in file.GetDirectories())
                        {
                            foreach (var file2 in file1.GetDirectories())
                            {
                                foreach (var file3 in file2.GetFiles())
                                {
                                    List<string> d = new List<string>();
                                    d.Add(file.Name); d.Add(file1.Name); d.Add(file2.Name);
d.Add(file3.Name);
                                    AllNames(d, dir, names, "nprop");
                                }
                            }
                        }
                    }
                    else if (file.Name != "documentation")
                    {
                        foreach (var file1 in file.GetDirectories())
                        {
                            foreach (var file2 in file1.GetFiles())
                            {
                                if (file2.Name.Contains(".exb"))
                                {
                                    List<string> d = new List<string>();
                                    d.Add(file.Name); d.Add(file1.Name); d.Add(file2.Name);
                                    AllNames(d, dir, names, "propr");
                                }
                            }
                        }
                    }
                }
            }
            int[] nodelete = { 15, 18, 21, 22, 24, 25, 26, 27, 28, 29, 30, 31, 34, 35, 37, 42,
44, 45, 46, 47, 48, 49, 52, 54, 55, 56, 57, 58, 61, 62, 63, 64, 65, 69, 79, 84, 89, 92, 93,
98, 99, 100, 101, 106, 115, 116, 127, 134, 141, 142, 143, 146, 147, 148, 149, 153, 154, 156,
159, 161, 162, 163, 175, 205, 207, 209, 215, 220, 224, 232, 234, 235, 237, 239, 240, 251, 252,
253, 259, 261, 265, 266, 270, 271, 273, 274, 285, 291, 293, 294, 295, 296, 301, 302, 304, 305,
307, 308, 309, 310, 314, 315, 318, 319, 325, 326, 329, 330, 332, 335, 338, 339, 340, 341, 342,
344, 345, 348, 354, 357, 358, 361, 362, 371, 372, 373, 374, 378, 379, 380, 383, 384, 385, 386,
387, 388, 389, 390, 391, 392, 406, 407, 408, 434, 438, 444, 447, 448, 452, 454, 455, 464, 471,
477, 478, 479, 480, 481, 482, 483, 484, 485, 487, 499, 503, 504, 505, 510, 512, 515, 520, 521,
522, 528, 530, 531, 532, 534, 535, 536, 539, 540, 549, 554, 555, 558, 559, 560, 563, 564, 573,
574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 586, 587, 588, 589, 590, 593, 599, 600,
601, 602, 609, 610, 611, 612, 614, 617, 622, 654, 655, 656, 659, 660, 661, 662, 663, 664, 665,
666, 668, 673, 681, 682, 683, 693, 699, 701, 704, 720, 727, 731, 732, 736, 739, 748, 749, 750,
765, 766, 767, 768, 771, 776, 777, 781, 785, 786, 787, 788, 789, 793, 794, 795, 805, 807, 814,
815, 816, 838, 841, 842, 846, 847, 848, 852, 853, 857, 862, 877, 878, 879, 880, 881, 882, 883,
884, 885, 886, 887, 891, 892, 896, 899, 901, 902, 904, 906, 907, 908, 914, 916, 917, 918, 920,
923, 924, 926, 932, 933, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 952, 953, 954, 960,
961, 967, 968, 969, 970, 971, 976, 977, 978, 979, 980, 981, 982, 983, 984, 990, 991, 992, 993,
995, 996, 999, 1000, 1003, 1004, 1005, 1009, 1010, 1012, 1013, 1014 };
            for (int i = names.Count - 1, j = nodelete.Length - 1; i >= 0; i--)
            {
                if (j < 0) names.RemoveAt(i);
                else
                {
                    if (i != nodelete[j]) names.RemoveAt(i);
                    else if (i == nodelete[j]) j--;
                }
            }
```

```
            XmlDocument namesInfo = new XmlDocument();
            namesInfo.Load("XMLFile1.xml");
            foreach (Names cons in names)
            {
                XmlElement highlightStyle = namesInfo.CreateElement("TextName");
                highlightStyle.SetAttribute("name", getText(cons.refname));
                namesInfo["Document"].AppendChild(highlightStyle);
                XmlElement PlaceNode = namesInfo.CreateElement("Place");
                highlightStyle.AppendChild(PlaceNode);
                if (cons.name == "Ленинград")
                    cons.name = "Санкт-Петербург";
                XmlText PlaceTxt = namesInfo.CreateTextNode(cons.name);
                PlaceNode.AppendChild(PlaceTxt);
                XmlElement DateNode = namesInfo.CreateElement("RecordDate");
                DateNode.SetAttribute("date", getDate(cons.refname));
                highlightStyle.AppendChild(DateNode);
                XmlElement SpeakerNode = namesInfo.CreateElement("Speaker");
                SpeakerNode.SetAttribute("speaker", cons.speaker);
                highlightStyle.AppendChild(SpeakerNode);
                XmlElement LanNode = namesInfo.CreateElement("Language");
                LanNode.SetAttribute("lang", cons.language);
                highlightStyle.AppendChild(LanNode);
            }
            namesInfo.Save("namesInfo.xml");
            Console.ReadLine();
        }
    }
}
```

Класс:

```
    public class Names
    {
        public string name { get; set; }            //название места
        public string start { get; set; }
        public string end { get; set; }
        public string refname { get; set; }        //название текста
        public int refnum { get; set; }              //номер предложения
        public string ts { get; set; }              //предложение в оригинале
        public string fr { get; set; }               //перевод предложения
        public string speaker { get; set; }          //кто записывал этот текст
        public string language { get; set; }
        public Names() { }
        public Names(string name, string refname, int refnum, string ts, string fr, string
speaker, string language)
        {
            this.name = name;
            this.refname = refname;
            this.refnum = refnum;
            this.ts = ts;
            this.fr = fr;
            this.speaker = speaker;
            this.language = language;
        }
    }
```

2) Изменение одинаковых координат для удобства просмотра

```
XmlDocument doc = new XmlDocument();
            doc.Load("AllMap.kml");
            XmlNodeList xnodeList = doc.GetElementsByTagName("Placemark");
```

```
            foreach (XmlNode xnode in xnodeList)
            {
                foreach (XmlNode node in xnode.ChildNodes)
                {
                    if(node.Name=="name")
                    {
                        name = node.InnerText;
                    }
                    if (node.Name == "styleUrl")
                    {
                        style = node.InnerText;
                    }
                    if (node.Name == "Style")
                    {
                        style = node.Attributes["id"].Value;
                    }
                    if (node.Name == "ExtendedData")
                    {
                        foreach(XmlNode el in node.ChildNodes)
                        {
                            if (el.Attributes["name"].Value == "titleEN")
                            {
                                titleEN = node.InnerText;
                            }
                            if (el.Attributes["name"].Value == "recordDate")
                            {
                                date = node.InnerText;
                            }
                            if (el.Attributes["name"].Value == "speakerName")
                            {
                                speaker = node.InnerText;
                            }
                        }
                    }
                    if (node.Name=="Point")
                    {
                        latitude = node["coordinates"].InnerText.Remove(0,
node["coordinates"].InnerText.IndexOfAny("0123456789".ToCharArray()));
                        longitude = latitude.Remove(0, latitude.IndexOf(",") + 1);
                        if (longitude.Contains(","))
                            longitude = longitude.Remove(longitude.IndexOf(","));
                        latitude = latitude.Remove(latitude.IndexOf(","));
                        lat = Convert.ToDouble(latitude.Replace(".", ","));
                        longi = Convert.ToDouble(longitude.Replace(".", ","));
                        foreach (XmlNode xxnode in xnodeList)
                        {
                            foreach (XmlNode nod in xxnode.ChildNodes)
                            {
                                if (nod.Name == "name")
                                {
                                    furname = nod.InnerText;
                                }
                                if (node.Name == "styleUrl")
                                {
                                    furstyle = node.InnerText;
                                }
                                if (node.Name == "Style")
                                {
                                    furstyle = node.Attributes["id"].Value;
                                }
                                if (node.Name == "ExtendedData")
                                {
                                    foreach (XmlNode el in node.ChildNodes)
                                    {
```

```
                                    if (el.Attributes["name"].Value == "titleEN")
                                    {
                                        furtitleEN = node.InnerText;
                                    }
                                    if (el.Attributes["name"].Value == "recordDate")
                                    {
                                        furdate = node.InnerText;
                                    }
                                    if (el.Attributes["name"].Value == "speakerName")
                                    {
                                        furspeaker = node.InnerText;
                                    }
                                }
                            }
                            if (furname != name|| titleEN!= furtitleEN|| date!= furdate||
speaker!= furspeaker|| style!= furstyle)
                            {
                                if (nod.Name == "Point")
                                {
                                    latitude = nod["coordinates"].InnerText.Remove(0,
nod["coordinates"].InnerText.IndexOfAny("0123456789".ToCharArray()));
                                    longitude = latitude.Remove(0, latitude.IndexOf(",") +
1);
                                    if(longitude.Contains(","))
                                        longitude =
longitude.Remove(longitude.IndexOf(","));
                                    latitude = latitude.Remove(latitude.IndexOf(","));
                                    furlat = Convert.ToDouble(latitude.Replace(".", ","));
                                    furlongi = Convert.ToDouble(longitude.Replace(".",
","));
                                    if (lat == furlat && longi == furlongi)
                                    {
                                        if (ll)
                                        {
                                            if (num == 1)
                                                furlongi += change;
                                            if (num == 2)
                                                furlat += change;
                                        }
                                        if (!ll)
                                        {
                                            if (num == 3)
                                                furlongi -= change;
                                            if (num == 4)
                                                furlat -= change;
                                        }
                                        if (num == 2)
                                        {
                                            ll = false;
                                        }
                                        if (num == 4)
                                        {
                                            change += 0.01;
                                            ll = true;
                                            num = 1;
                                        }
                                        else num++;
                                        string txt = String.Format("{0},{1}",
furlat.ToString().Replace(",", "."), furlongi.ToString().Replace(",", "."));
                                        nod["coordinates"].InnerText = txt;
                                    }
                                }
                            }
                        }
```

```
                    }
                }
                num = 1;
                ll = true;
            }
        }
        doc.Save("allMap.kml");
```

3) Добавление путей в файлы с координатами трёх языков

```
        public static void Create(XmlDocument Map, XmlNode xnode, string coText, string
xcoText, string speaker)
        {
            XmlElement NewX = Map.CreateElement("Placemark");
            Map["kml"]["Document"].AppendChild(NewX);

            XmlElement nameX = Map.CreateElement("name");
            NewX.AppendChild(nameX);
            XmlText nameText = Map.CreateTextNode(speaker);
            nameX.AppendChild(nameText);

            XmlElement styleNode = Map.CreateElement("styleUrl");
            NewX.AppendChild(styleNode).AppendChild(Map.CreateTextNode("#sel"));

            XmlElement lineNode = Map.CreateElement("LineString");
            NewX.AppendChild(lineNode);

            XmlElement exNode = Map.CreateElement("extrude");
            lineNode.AppendChild(exNode).AppendChild(Map.CreateTextNode("1"));
            XmlElement teNode = Map.CreateElement("tessellate");
            lineNode.AppendChild(teNode).AppendChild(Map.CreateTextNode("1"));
            XmlElement alNode = Map.CreateElement("altitudeMode");
            lineNode.AppendChild(alNode).AppendChild(Map.CreateTextNode("clampToGround"));
            XmlElement coNode = Map.CreateElement("coordinates");
            lineNode.AppendChild(coNode);
            coNode.AppendChild(Map.CreateTextNode(xcoText));
            coNode.AppendChild(Map.CreateTextNode(" "));
            coNode.AppendChild(Map.CreateTextNode(coText));
        }

XmlDocument doc = new XmlDocument();
            XmlText coText = doc.CreateTextNode(""), xcoText = doc.CreateTextNode("");
            doc.Load("sel_link.kml");
            XmlNodeList xnodeList = doc.GetElementsByTagName("Placemark");

            XmlDocument Map = new XmlDocument();
            Map.Load("start.xml");

            foreach (XmlNode xnode in xnodeList)
3            {
                foreach (XmlNode node in xnode.ChildNodes)
                {
                    if (node.Name == "Style" && node.Attributes["id"].Value ==
"highlightPlacemark")
                        ll = true;
                    if (node.Name == "Point" && ll)
                        coTexts = node["coordinates"].InnerText;
                    if (node.Name == "name" )
                        name = node.InnerText;
                    if (node.Name == "ExtendedData" && ll)
                    {
                        foreach (XmlNode el in node.ChildNodes)
                        {
```

```
                            if (el.Attributes["name"].Value == "Type") ll = false;
                            if (el.Attributes["name"].Value == "speakers")
                                speaker = el.InnerText;
                        }
                    }
                }
                if (ll)
                {
                    foreach (XmlNode nod in xnodeList)
                    {
                        foreach (XmlNode no in nod.ChildNodes)
                        {
                            if (no.Name == "styleUrl" && no.InnerText == "#namesPlacemark")
                                notxt = true;  //если связываются название текста и ГО (из
.exb файлов)
                            if (no.Name == "Point")
                                xcoTexts = no["coordinates"].InnerText;
                            if (no.Name == "ExtendedData")
                            {
                                foreach (XmlNode el in no.ChildNodes)
                                {
                                    //if (el.Attributes["name"].Value == "Type") notxt = true;
//если связываются название текста и те места, которые связаны с автором
                                    if (el.Attributes["name"].Value == "speakers")
                                        furspeaker = el.InnerText;
                                    if (el.Attributes["name"].Value == "TitleText")
                                        furname = el.InnerText;
                                }
                            }
                        }
                        if (name == furname && coTexts != xcoTexts && furspeaker != "" &&
notxt)
                        {
                            Create(Map, xnode, coTexts, xcoTexts, speaker);
                        }
                        notxt = false;
                    }
                }
                ll = false;
            }
            Map.Save("sel_link.kml");
```

4) HTML

```
<!DOCTYPE html>

<html>

    <head>

        <title>Визуализация</title>

        <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css" integri
ty="sha512-
xwE/Az9zrjBIphAcBb3F6JVqxf46+CDLwfLMHloNu6KEQCAWi6HcDUbeOfBIptF7tcCzusKFjFw2yuvEpDL9wQ==" cros
sorigin=""/>


         <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"

            integrity="sha512-
gZwIG9x3wUXg2hdXF6+rVkLF/0Vi9U8D2Ntg4Ga5I5BZpVkVxlJWbSQtXPSiUTtC0TjtGOmxa1AJPuV0CPthew=="

            crossorigin=""></script>
```

```
            <link rel="stylesheet" type="text/css" href="style.css">
            <script src="https://kit.fontawesome.com/b6eda2fd05.js" crossorigin="anonymous"></
script>
            <script src="map_js.js"></script>

    </head>
    <body onload="output_dlg(), output_xas(), output_sel()">
        <header>
            <form>
                <input type="text" id="elastic" name="search" placeholder="Search" >
                <button class="fa fa-search" type="submit" id="demo"></button>
            </form>
        </header>

        <input type="checkbox" id="check">
        <label for="check">
            <i class="fas fa-angle-right" id="btn"></i> </i>
            <i class="fas fa-angle-left" id="cancel"> </i>
        </label>

        <div class="sidebar">
            <ul>
                <li><a href="#" style="text-align: center; color: #ffff;">Search</a></li>
                <li>
                    <a href="#">
                        <p><b>Languages:</b></p>
                        <input type="checkbox" id="lang1" value="Dolgan" onchange="output_dlg(
)" checked>
                        <label for="lang1">Dolgan</label>
                        <input type="checkbox" id="lang2" value="Selkup" onchange="output_sel(
)" checked>
                        <label for="lang2">Selkup</label>
                        <input type="checkbox" id="lang3" value="Kamas" onchange="output_xas()
" checked>
                        <label for="lang3">Kamas</label>
                    </a>
                </li>
                <li>
```

```
                    <a href="#">

                        <p><b>Year of record: </b><output for="YearRange1" id="year1">1912</ou
tput> – <output for="YearRange2" id="year2">2019</output></p>

                    </a>

                        <div class="slidecontainer">

                            <input type="range" id="YearRange1" max="2019" min="1912"  value="
1912" step="1" oninput="outputUpdateYear1(value)" onchange="output_dlg(), output_xas(), output
_sel()">

                            <input type="range" id="YearRange2" max="2019" min="1912"  value="
2019" step="1" oninput="outputUpdateYear2(value)" onchange="output_dlg(), output_xas(), output
_sel()">

                        </div>


                </li>

                <li>

                    <a href="#">

                        <b>Sex:</b>

                        <input type="checkbox" id="sex1" value="Male" checked onchange="output
_dlg(), output_xas(), output_sel()">

                        <label for="sex1">Male</label>

                        <input type="checkbox" id="sex2" value="Female" checked onchange="outp
ut_dlg(), output_xas(), output_sel()">

                        <label for="sex2">Female</label>

                    </a>

                </li>

                <li>

                    <a href="#">

                        <p><b>Age of speaker: </b><output for="AgeRange1" id="age1">10</output
> – <output for="AgeRange1" id="age2">136</output></p>

                    </a>

                        <div class="slidecontainer" style="display: block;">

                            <input type="range" id="AgeRange1" max="100" min="10"  value="10"
step="1" oninput="outputUpdateAge1(value)" onchange="output_dlg(), output_xas(), output_sel()"
>

                            <input type="range" id="AgeRange2" max="136" min="10"  value="136"
 step="1" oninput="outputUpdateAge2(value)" onchange="output_dlg(), output_xas(), output_sel()
">

                        </div>

                        <input type="checkbox" id="agenone" value="" checked onchange="output_
dlg(), output_xas(), output_sel()">

                        <label for="agenone"><b>Show people without a known year of birth</b><
/label>

                </li>
```

```
                <li>
                    <a href="#">
                        <p><b>Paths: </b></p>
                        <label class="switch">
                            <input type="checkbox" id="check_switch" onchange="output_dlg(), o
utput_xas(), output_sel()">
                            <span class="slider round"></span>
                        </label>
                    </a>
                </li>
            </ul>
        </div>

        <section id="mapid"></section>

    <script src = "scripts.js"></script>
    </body>
</html>
```

5) CSS

```
#mapid{
    height: 100vmin;
    width: 100vmax;
 }

 input:focus{
    outline: none;
 }

 section{
    position: fixed;
    z-index: -1;
}

 #save{
        background-color: #4CAF50; /* Green */
        border: none;
        color: white;
```

```
        padding: 20px 40px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
 }

 #elastic{
     width: 330px;
     height: 45px;
     margin-left: 35px;
     border: none;
     border-radius: 4px;
     user-select: none;
     box-shadow: 0px 0px 5px darkgrey;
 }

header{
    position: fixed;
    z-index: 1000;
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
    font-size: 26px;
    left: -24px;
    text-align: left;
    line-height: 50px;
    user-select: none;
}

button {
  position: absolute;
  top: 0;
  right: 0px;
  width: 42px;
  height: 42px;
  border: none;
  cursor: pointer;
}
```

```
button:before {
  content: "\f002";
  font-family: FontAwesome;
  font-size: 16px;
  color: rgb(0, 0, 0);
}

* {box-sizing: border-box;}
form {
  position: relative;
  width: 300px;
  margin: 0 auto;
}

input, button {
  border: none;
  outline: none;
  border-radius: 3px;
}

input {
  width: 100%;
  height: 42px;
  padding-left: 15px;
}

button {
  height: 28px;
  width: 28px;
  position: absolute;
  top: 15px;
  left: 325px;
  cursor: pointer;
  background-color: #ffffff;
}

button:before {
```

```
  font-family: FontAwesome;
  color: rgb(0, 0, 0);
  font-size: 20px;
  font-weight: bold;
}

 *{
    margin: 0;
    padding: 0;
    list-style: none;
    text-decoration: none;
 }

 .sidebar{
    position: fixed;
    z-index: 101;
    left: -350px;
    width: 350px;
    height: 100%;
    background: #ffffff;
    transition: all .5s ease;
    box-shadow: 0px 0px 5px darkgrey;
 }

.sidebar ul a{
    display: block;
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
    height: 100%;
    width: 100%;
    font-size: 18px;
    color: black;
    text-align: left;
    line-height: 55px;
    padding-left: 30px;
    box-sizing: border-box;
    border-top: 1px solid #cccccc;
    transition: .3s;
```

```
}

/*.sidebar li:hover a{
    background-color: rgba(236, 236, 236, 0.8);
}*/

#check{
    display: none;
}

label #btn, label #cancel{
    position: absolute;
    cursor: pointer;
    background: rgba(0, 0, 0, 0.445);
}

label #btn{
    z-index: 1000;
    left: 350px;
    top: 16px;
    font-size: 25px;
    color: rgba(255, 255, 255, 0.76);
    padding: 4px 8px;
    border-radius: 3px;
    transition: all .5s;
}

label #cancel{
    z-index: 1000;
    left: -355px;
    top: 16px;
    font-size: 25px;
    color: rgba(255, 255, 255, 0.76);
    padding: 4px 8px;
    border-radius: 3px;
    transition: all .5s ease;
}
```

```
#check:checked ~ .sidebar{
    left: 0;
}

#check:checked ~ label #btn{
    left: 340px;
    opacity: 0;
    pointer-events: none;
}

#check:checked ~ label #cancel{
    left: 350px;
}

#lang{
    margin-left: 15px;
}

label{
    cursor: pointer;
    color: #000;
    font-weight: normal;
    line-height: 30px;
    padding: 10px 0;
}

input[type=checkbox] {
    display: none;
}

label[for="lang1"]::before, label[for="lang2"]::before, label[for="lang3"]::before, label[for=
"sex1"]::before, label[for="sex2"]::before {

  content: " ";
  display: block;
  color: #000;
  display: inline-block;
```

```
  font: 20px/30px Arial;
  margin-right: 5px;
  margin-left: 15px;
  position: relative;
  text-align: center;
  text-indent: 0px;
  width: 11px;
  height: 11px;
  background: #FFF;
  border: 1px solid #e3e3e3;
  border-image: initial;
}

label[for="agenone"]::before{
  content: " ";
  color: #000;
  display: inline-block;
  font: 20px/30px Arial;
  margin-right: 5px;
  margin-left: 15px;
  position: relative;
  text-align: center;
  text-indent: 0px;
  width: 11px;
  height: 11px;
  background: #FFF;
  border: 1px solid #e3e3e3;
  border-image: initial
}

input:checked + label[for="lang1"]::before, input:checked + label[for="lang2"]::before, input:
checked + label[for="lang3"]::before {
    content: "\2713";
    text-shadow: 1px 1px 1px rgba(0, 0, 0, 0.541);
    font-size: 15px;
    color: #000000;
    text-align: center;
    line-height: 15px;
```

```
}

input:checked + label[for="sex1"]::before, input:checked + label[for="sex2"]::before{
    content: "\2713";
    text-shadow: 1px 1px 1px rgba(0, 0, 0, 0.541);
    font-size: 15px;
    color: #000000;
    text-align: center;
    line-height: 15px;
}

input:checked + label[for="agenone"]::before{
  position: relative;
  content: "\2713";
  text-shadow: 1px 1px 1px rgba(0, 0, 0, 0.541);
  font-size: 15px;
  color: #000000;
  text-align: center;
  line-height: 15px;
  display: inline-block;
}

.slidecontainer{
    position: relative;
    user-select: none;
    pointer-events: none;
    border-radius: 8px;
    width: 290px;
    height: 25px;
    margin-bottom: 5px;
    left: 25px;
}

output #YearRange1, output #AgeRange1{
    position: absolute;
    margin-left: 15px;
    border-radius: 8px 8px 0px 8px;
```

```
    font-weight: bold;
    color: #020202;
}

output #YearRange2, output #AgeRange2{
    position: absolute;
    margin-left: 30px;
    font-weight: bold;
    color: #020202;
}

#YearRange1, #AgeRange1{
    width: 280px;
    height: 7px;
    left: 5px;
    bottom: 16px;
    background: #75cff0;
    -webkit-appearance: none;
    outline: none;
    border-radius: 4px;
}

#YearRange2, #AgeRange2{
    background: #0a95da00;
    width: 280px;
    height: 7px;
    left: 5px;
    bottom: 16px;
    -webkit-appearance: none;
    outline: none;
    border-radius: 4px;
}

.slidecontainer input[type="range"]{
    height: 2em;
    display: block;
    margin: 0;
```

```
    padding: 0;
  }

.slidecontainer input[type="range"]:last-child{
    margin-top: -0.56em;
}

input[type=range]::-webkit-slider-thumb {
    -webkit-appearance: none;
    pointer-events: all;
    background-color: #67bedd;
    width: 20px;
    height: 20px;
    border-radius: 10px;
    cursor: pointer;
    margin-top: -4px;
    user-select: none;
    transition: .5s;
}

.switch {
  position: relative;
  display: inline-block;
  width: 40px;
  height: 34px;
}

.switch input {display:none;}

.slider {
  position: absolute;
  cursor: pointer;
  width: 60px;
  top: -45px;
  left: 64px;
  right: 0;
  bottom: 45px;
```

```
  background-color: #ccc;
  -webkit-transition: .4s;
  transition: .4s;
}

.slider:before {
  position: absolute;
  content: "";
  height: 26px;
  width: 26px;
  left: 4px;
  bottom: 4px;
  background-color: white;
  -webkit-transition: .4s;
  transition: .4s;
}

input:checked + .slider {
  background-color: #67bedd;
}

input:focus + .slider {
  box-shadow: 0 0 1px #67bedd;
}

input:checked + .slider:before {
  -webkit-transform: translateX(26px);
  -ms-transform: translateX(26px);
  transform: translateX(26px);
}

.slider.round {
  border-radius: 34px;
}

.slider.round:before {
  border-radius: 50%;
```

```
}
```

6) JavaScript

```
const mymap = L.map('mapid').setView([67, 83], 4);

mymap.zoomControl.setPosition('topright');


let dlg_layer = L.layerGroup();

let sel_layer = L.layerGroup();

let xas_layer = L.layerGroup();

let path_dlg = L.layerGroup();

let path_sel = L.layerGroup();

let path_xas = L.layerGroup();


L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token=pk.eyJ1Ijoib
WFwYm94IiwiYSI6ImNpejY4NXVycTA2emYycXBndHRqcmZ3N3gifQ.rJcFIG214AriISLbB6B5aw', {

minZoom: 3,

maxZoom: 20,

attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contr
ibutors, ' +

'<a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, ' +

'Imagery © <a href="https://www.mapbox.com/">Mapbox</a>',

id: 'mapbox/streets-v11',

tileSize: 512,

zoomOffset: -1

}).addTo(mymap);


/*fetch('assets/INEL_LangsRecolored.kml')

    .then(res => res.text())

    .then(kmltext => {

        // Create new kml overlay

        const parser = new DOMParser();

        const kml = parser.parseFromString(kmltext, 'text/xml');

        const track = new L.KML(kml);

        mymap.addLayer(track);


        // Adjust map to show the kml

        const bounds = track.getBounds();

        mymap.fitBounds(bounds);
```

```
    });*/

let m_layer = L.layerGroup();
let f_layer = L.layerGroup();
let unknown_year = L.layerGroup();

function output_dlg(num){
    dlg_layer.clearLayers();
    let parser, xmlDoc;
    var layer = [], l, path = [];
    let check = document.getElementById('check_switch');
    let checkbox_lang = document.getElementById("lang1");
    let checkbox_male = document.getElementById("sex1");
    let checkbox_female = document.getElementById("sex2");
    let checkbox_unknown = document.getElementById('agenone');

    if(checkbox_lang.checked){
        fetch('assets/allKML.kml')
        .then(res => res.text())
        .then(kmltext => {
            parser = new DOMParser();
            xmlDoc = parser.parseFromString(kmltext, 'text/xml');

            var style = L.KML.parseStyles(xmlDoc);
            L.KML.parseStyleMap(xmlDoc, style);

            let list_points = xmlDoc.getElementsByTagName('Placemark');

            for(let i = 0; i < list_points.length; i++){
                let help = list_points[i].getElementsByTagName('Data')[0].childNodes[1].childN
odes[0].nodeValue;
                if(help != 'Path'){
                    if(help == "Text"){
                        let year_help = list_points[i].getElementsByTagName('Data')[4].childNo
des[1].childNodes[0].nodeValue;
                        if(year_help != '...' && list_points[i].getElementsByTagName('Data')[5
].childNodes[1].childNodes[0].nodeValue == 'dlg'){
                            if(+year_help > +sliders[0].value && +year_help < +sliders[1].valu
e){
```

```
                                l = L.KML.parsePlacemark(list_points[i], xmlDoc, style);
                                if(l) { layer.push(l); }
                            }
                        }
                    }
                    else{
                        if(help == "From Text"){
                            let year_help = list_points[i].getElementsByTagName('Data')[4].chi
ldNodes[1].childNodes[0].nodeValue;
                            if(year_help != '...' && list_points[i].getElementsByTagName('Data
')[3].childNodes[1].childNodes[0].nodeValue == 'dlg'){
                                if(+year_help > +sliders[0].value && +year_help < +sliders[1].
value){
                                    l = L.KML.parsePlacemark(list_points[i], xmlDoc, style);
                                    if(l) { layer.push(l); }
                                }
                            }
                        }

                        if(help == 'Place of birth' || help == 'Domicile' || help == 'Former r
esidence'){

                            if(checkbox_male.checked){
                                let year_help = list_points[i].getElementsByTagName('Data')[7]
.childNodes[1].childNodes[0].nodeValue;
                                lang_help = list_points[i].getElementsByTagName('Data')[3].chi
ldNodes[1].childNodes[0].nodeValue;
                                sex_help = list_points[i].getElementsByTagName('Data')[4].chil
dNodes[1].childNodes[0].nodeValue;
                                if( sex_help == 'male' && lang_help == 'dlg'){
                                    if(year_help != '...' && +year_help > +sliders[2].value &&
 +year_help < +sliders[3].value)
                                    {
                                        l = L.KML.parsePlacemark(list_points[i], xmlDoc, style
);
                                        if(l) { layer.push(l); }
                                    }
                                    else{
                                        if(checkbox_unknown.checked){
                                            l = L.KML.parsePlacemark(list_points[i], xmlDoc, s
tyle);
                                            if(l) { layer.push(l); }
```

```
                                    for(let i = 0; i < layer.length; i++){
                                        unknown_year.addLayer(layer[i]);
                                    }
                                    unknown_year.addTo(mymap);
                                }
                                else{
                                    unknown_year.clearLayers();
                                }
                            }
                        }
                        for(let i = 0; i < layer.length; i++){
                            m_layer.addLayer(layer[i]);
                        }
                        m_layer.addTo(mymap);
                    }
                    else{
                        m_layer.clearLayers();
                    }

                    if(checkbox_female.checked){
                        let year_help = list_points[i].getElementsByTagName('Data')[7]
.childNodes[1].childNodes[0].nodeValue;
                        lang_help = list_points[i].getElementsByTagName('Data')[3].chi
ldNodes[1].childNodes[0].nodeValue;
                        sex_help = list_points[i].getElementsByTagName('Data')[4].chil
dNodes[1].childNodes[0].nodeValue;
                        if( sex_help == 'female' && lang_help == 'dlg'){
                            if(year_help != '...' && +year_help > +sliders[2].value &&
 +year_help < +sliders[3].value)
                            {
                                l = L.KML.parsePlacemark(list_points[i], xmlDoc, style
);
                                if(l) { layer.push(l); }
                            }
                            else{
                                if(checkbox_unknown.checked){
                                    l = L.KML.parsePlacemark(list_points[i], xmlDoc, s
tyle);
                                    if(l) { layer.push(l); }
```

```
                                    for(let i = 0; i < layer.length; i++){
                                        unknown_year.addLayer(layer[i]);
                                    }
                                    unknown_year.addTo(mymap);
                                }
                                else{
                                    unknown_year.clearLayers();
                                }
                            }
                            for(let i = 0; i < layer.length; i++){
                                f_layer.addLayer(layer[i]);
                            }
                            f_layer.addTo(mymap);
                        }
                        else{
                            f_layer.clearLayers();
                        }
                    }
                }
                else{
                    if(check.checked){
                        if(list_points[i].getElementsByTagName('styleUrl')[0].childNodes[0].no
deValue == '#dlg'){
                            l = L.KML.parsePlacemark(list_points[i], xmlDoc, style);
                            if(l) { path.push(l); }
                        }
                        for(let i = 0; i < path.length; i++){
                            path_dlg.addLayer(path[i]);
                        }
                        path_dlg.addTo(mymap);
                    }
                    else{
                        path_dlg.clearLayers();
                    }
                }
```

```
            };
            for(let i = 0; i < layer.length; i++){
                dlg_layer.addLayer(layer[i]);
            }
            dlg_layer.addTo(mymap);
        });
    }
    else{
        dlg_layer.clearLayers();
        path_dlg.clearLayers();
    }
}

function output_xas(num){
    xas_layer.clearLayers();
    let parser, xmlDoc;
    var layer = [], l, path = [];
    let check = document.getElementById('check_switch');
    let checkbox_lang = document.getElementById("lang3");
    let checkbox_male = document.getElementById("sex1");
    let checkbox_female = document.getElementById("sex2");
    let checkbox_unknown = document.getElementById("agenone");

    if(checkbox_lang.checked){
        fetch('assets/allKML.kml')
        .then(res => res.text())
        .then(kmltext => {
            parser = new DOMParser();
            xmlDoc = parser.parseFromString(kmltext, 'text/xml');

            var style = L.KML.parseStyles(xmlDoc);
            L.KML.parseStyleMap(xmlDoc, style);

                let list_points = xmlDoc.getElementsByTagName('Placemark');

                for(let i = 0; i < list_points.length; i++){
                    let help = list_points[i].getElementsByTagName('Data')[0].childNodes[1].ch
ildNodes[0].nodeValue;
```

```
                    if(help != 'Path'){
                        if(help == "Text"){
                            let year_help = list_points[i].getElementsByTagName('Data')[4].chi
ldNodes[1].childNodes[0].nodeValue;
                            if(year_help != '...' && list_points[i].getElementsByTagName('Data
')[5].childNodes[1].childNodes[0].nodeValue == 'xas'){
                                if(+year_help > +sliders[0].value && +year_help < +sliders[1].
value){
                                    l = L.KML.parsePlacemark(list_points[i], xmlDoc, style);
                                    if(l) { layer.push(l); }
                                }
                            }
                        }
                        else{
                            if(help == "From Text"){
                                let year_help = list_points[i].getElementsByTagName('Data')[3]
.childNodes[1].childNodes[0].nodeValue;
                                if(year_help != '...' && list_points[i].getElementsByTagName('
Data')[4].childNodes[1].childNodes[0].nodeValue == 'xas'){
                                    if(+year_help > +sliders[0].value && +year_help < +sliders
[1].value){
                                        l = L.KML.parsePlacemark(list_points[i], xmlDoc, style
);
                                        if(l) { layer.push(l); }
                                    }
                                }
                            }


                            if(help == 'Place of birth' || help == 'Domicile' || help == 'Form
er residence'){


                                if(checkbox_male.checked){
                                    let year_help = list_points[i].getElementsByTagName('Data'
)[7].childNodes[1].childNodes[0].nodeValue;
                                        lang_help = list_points[i].getElementsByTagName('Data'
)[3].childNodes[1].childNodes[0].nodeValue;
                                        sex_help = list_points[i].getElementsByTagName('Data')
[4].childNodes[1].childNodes[0].nodeValue;
                                        if( sex_help == 'male' && lang_help == 'xas'){
                                            if(year_help != '...' && +year_help > +sliders[2].
value && +year_help < +sliders[3].value)
                                            {
```

```
                                        l = L.KML.parsePlacemark(list_points[i], xmlDo
c, style);
                                        if(l) { layer.push(l); }
                                    }
                                    else{
                                        if(checkbox_unknown.checked){
                                            l = L.KML.parsePlacemark(list_points[i], x
mlDoc, style);
                                            if(l) { layer.push(l); }

                                            for(let i = 0; i < layer.length; i++){
                                                unknown_year.addLayer(layer[i]);
                                            }
                                            unknown_year.addTo(mymap);
                                        }
                                        else{
                                            unknown_year.clearLayers();
                                        }
                                    }
                                }

                            for(let i = 0; i < layer.length; i++){
                                m_layer.addLayer(layer[i]);
                            }
                            m_layer.addTo(mymap);
                        }
                        else{
                            m_layer.clearLayers();
                        }

                        if(checkbox_female.checked){
                            let year_help = list_points[i].getElementsByTagName('Data'
)[7].childNodes[1].childNodes[0].nodeValue;
                            lang_help = list_points[i].getElementsByTagName('Data')[3]
.childNodes[1].childNodes[0].nodeValue;
                            sex_help = list_points[i].getElementsByTagName('Data')[4].
childNodes[1].childNodes[0].nodeValue;
                            if( sex_help == 'female' && lang_help == 'xas'){
                                if(year_help != '...' && +year_help > +sliders[2].valu
e && +year_help < +sliders[3].value)
```

```
                                        {
                                            l = L.KML.parsePlacemark(list_points[i], xmlDoc, s
tyle);
                                            if(l) { layer.push(l); }
                                        }
                                        else{
                                            if(checkbox_unknown.checked){
                                                l = L.KML.parsePlacemark(list_points[i], xmlDo
c, style);
                                                if(l) { layer.push(l); }

                                                for(let i = 0; i < layer.length; i++){
                                                    unknown_year.addLayer(layer[i]);
                                                }
                                                unknown_year.addTo(mymap);
                                            }
                                            else{
                                                unknown_year.clearLayers();
                                            }
                                        }
                                    }
                                    for(let i = 0; i < layer.length; i++){
                                        f_layer.addLayer(layer[i]);
                                    }
                                    f_layer.addTo(mymap);
                                }
                                else{
                                    f_layer.clearLayers();
                                }
                            }
                        }
                    }
                    else{
                        if(check.checked){
                            if(list_points[i].getElementsByTagName('styleUrl')[0].childNodes[0
].nodeValue == '#xas'){
                                l = L.KML.parsePlacemark(list_points[i], xmlDoc, style);
                                if(l) { path.push(l); }
                            }
```

```
                            for(let i = 0; i < path.length; i++){
                                path_xas.addLayer(path[i]);
                            }
                            path_xas.addTo(mymap);
                        }
                        else{
                            path_xas.clearLayers();
                        }
                    }
                };
                for(let i = 0; i < layer.length; i++){
                    xas_layer.addLayer(layer[i]);
                }
                xas_layer.addTo(mymap);
        });
    }
    else{
        xas_layer.clearLayers();
        path_xas.clearLayers();
    }
}

function output_sel(num){
    sel_layer.clearLayers();
    let parser, xmlDoc;
    var layer = [], l, path = [];
    let check = document.getElementById('check_switch');
    let checkbox_lang = document.getElementById("lang2");
    let checkbox_male = document.getElementById("sex1");
    let checkbox_female = document.getElementById("sex2");
    let checkbox_unknown = document.getElementById("agenone");

    if(checkbox_lang.checked){
        fetch('assets/allKML.kml')
        .then(res => res.text())
        .then(kmltext => {
            parser = new DOMParser();
```

```
            xmlDoc = parser.parseFromString(kmltext, 'text/xml');


            var style = L.KML.parseStyles(xmlDoc);
            L.KML.parseStyleMap(xmlDoc, style);


                let list_points = xmlDoc.getElementsByTagName('Placemark');


                for(let i = 0; i < list_points.length; i++){
                    let help = list_points[i].getElementsByTagName('Data')[0].childNodes[1].ch
ildNodes[0].nodeValue;
                    if(help != 'Path'){
                        if(help == "Text"){
                            let year_help = list_points[i].getElementsByTagName('Data')[4].chi
ldNodes[1].childNodes[0].nodeValue;
                            if(year_help != '...' && list_points[i].getElementsByTagName('Data
')[5].childNodes[1].childNodes[0].nodeValue == 'sel'){
                                if(+year_help > +sliders[0].value && +year_help < +sliders[1].
value){
                                    l = L.KML.parsePlacemark(list_points[i], xmlDoc, style);
                                    if(l) { layer.push(l); }
                                }
                            }
                        }
                        else{
                            if(help == "From Text"){
                                let year_help = list_points[i].getElementsByTagName('Data')[4]
.childNodes[1].childNodes[0].nodeValue;
                                if(year_help != '...' && list_points[i].getElementsByTagName('
Data')[3].childNodes[1].childNodes[0].nodeValue == 'sel'){
                                    if(+year_help > +sliders[0].value && +year_help < +sliders
[1].value){
                                        l = L.KML.parsePlacemark(list_points[i], xmlDoc, style
);
                                        if(l) { layer.push(l); }
                                    }
                                }
                            }


                            if(help == 'Place of birth' || help == 'Domicile' || help == 'Form
er residence'){
```

```
                                if(checkbox_male.checked){
                                    let year_help = list_points[i].getElementsByTagName('Data'
)[7].childNodes[1].childNodes[0].nodeValue;
                                        lang_help = list_points[i].getElementsByTagName('Data'
)[3].childNodes[1].childNodes[0].nodeValue;
                                        sex_help = list_points[i].getElementsByTagName('Data')
[4].childNodes[1].childNodes[0].nodeValue;
                                        if( sex_help == 'male' && lang_help == 'sel'){
                                            if(year_help != '...' && +year_help > +sliders[2].
value && +year_help < +sliders[3].value)
                                            {
                                                l = L.KML.parsePlacemark(list_points[i], xmlDo
c, style);
                                                if(l) { layer.push(l); }
                                            }
                                            else{
                                                if(checkbox_unknown.checked){
                                                    l = L.KML.parsePlacemark(list_points[i], x
mlDoc, style);
                                                    if(l) { layer.push(l); }

                                                    for(let i = 0; i < layer.length; i++){
                                                        unknown_year.addLayer(layer[i]);
                                                    }
                                                    unknown_year.addTo(mymap);
                                                }
                                                else{
                                                    unknown_year.clearLayers();
                                                }
                                            }
                                        }
                                    for(let i = 0; i < layer.length; i++){
                                        m_layer.addLayer(layer[i]);
                                    }
                                    m_layer.addTo(mymap);
                                }
                                else{
                                    m_layer.clearLayers();
                                }
```

```
                                if(checkbox_female.checked){
                                    let year_help = list_points[i].getElementsByTagName('Data'
)[7].childNodes[1].childNodes[0].nodeValue;
                                    lang_help = list_points[i].getElementsByTagName('Data')[3]
.childNodes[1].childNodes[0].nodeValue;
                                    sex_help = list_points[i].getElementsByTagName('Data')[4].
childNodes[1].childNodes[0].nodeValue;
                                    if( sex_help == 'female' && lang_help == 'sel'){
                                        if(year_help != '...' && +year_help > +sliders[2].valu
e && +year_help < +sliders[3].value)
                                        {
                                            l = L.KML.parsePlacemark(list_points[i], xmlDoc, s
tyle);
                                            if(l) { layer.push(l); }
                                        }
                                        else{
                                            if(checkbox_unknown.checked){
                                                l = L.KML.parsePlacemark(list_points[i], xmlDo
c, style);
                                                if(l) { layer.push(l); }

                                                for(let i = 0; i < layer.length; i++){
                                                    unknown_year.addLayer(layer[i]);
                                                }
                                                unknown_year.addTo(mymap);
                                            }
                                            else{
                                                unknown_year.clearLayers();
                                            }
                                        }
                                    }
                                    for(let i = 0; i < layer.length; i++){
                                        f_layer.addLayer(layer[i]);
                                    }
                                    f_layer.addTo(mymap);
                                }
                                else{
                                    f_layer.clearLayers();
                                }
                            }
```

```
                        }
                    }
                    else{
                        if(check.checked){
                            if(list_points[i].getElementsByTagName('styleUrl')[0].childNodes[0
].nodeValue == '#sel'){
                                l = L.KML.parsePlacemark(list_points[i], xmlDoc, style);
                                if(l) { path.push(l); }
                            }
                            for(let i = 0; i < path.length; i++){
                                path_sel.addLayer(path[i]);
                            }
                            path_sel.addTo(mymap);
                        }
                        else{
                            path_sel.clearLayers();
                        }
                    }
                };
                for(let i = 0; i < layer.length; i++){
                    sel_layer.addLayer(layer[i]);
                }
                sel_layer.addTo(mymap);
        });
    }
    else{
        sel_layer.clearLayers();
        path_sel.clearLayers();
    }
}

function outputUpdateYear1(year_v){
    let output = document.getElementById('year1');
    output.value = year_v;
    output.style.right = year_v;
}

function outputUpdateYear2(year_v){
```

```
    let output = document.getElementById('year2');
    output.value = year_v;
    output.style.right = year_v;
}

function outputUpdateAge1(age_v){
    let output = document.getElementById('age1');
    output.value = age_v;
    output.style.right = age_v;
}

function outputUpdateAge2(age_v){
    let output = document.getElementById('age2');
    output.value = age_v;
    output.style.right = age_v;
}

const sliders = document.querySelectorAll('input[type="range"]');

sliders[0].addEventListener('input', (e) => {
 if(+sliders[0].value > +sliders[1].value){
    sliders[1].value = +sliders[0].value;
  }
});

sliders[1].addEventListener('input', (e) => {
 if(+sliders[1].value < +sliders[0].value){
    sliders[0].value = +sliders[1].value;
  }
});

sliders[2].addEventListener('input', (e) => {
    if(+sliders[2].value > +sliders[3].value){
       sliders[3].value = +sliders[2].value;
     }
   });
```

```
sliders[3].addEventListener('input', (e) => {
if(+sliders[3].value < +sliders[2].value){
    sliders[2].value = +sliders[3].value;
    }
});
```