Санкт-Петербургский государственный университет

Математико-Механический факультет
Кафедра системного программирования

Немчинов Егор Игоревич

Разработка программы для изучения иностранных языков при просмотре фильмов

Курсовая работа

Научный руководитель:
ст. преп. Сартасов С. Ю.

Санкт-Петербург
2019

Оглавление

Введение

Знание иностранных языков открывает множество возможностей для людей, позволяя им путешествовать, работать за границей, узнавать новую культуру и устанавливать контакты с жителями других стран. По оценкам, половина людей или больше знают как минимум два языка [8]. Какие подходы существуют для изучения нового языка?

С приходом технологий методы изучения языков улучшились, эффективность повысилась, а обучение подстраивается под человека. Например, теперь человек в удобной игровой форме может узнавать новые слова и повторять их — ему в первую очередь дают те слова, которые он помнит хуже всего. Методы изучения языков заметно расширились с возможностями использования компьютеров для этого: *CALL, Computer-Assisted Language Learning*, [1]. Особенно эффективный метод интервального повторения [12] — техника запоминания, заключающаяся в повторении запоминаемого учебного материала по постоянно возрастающим интервалам — стал более доступен, т.к. программы могут планировать изучение для максимальной эффективности.

Полное погружение в среду иностранного языка — это эффективный способ изучения, однако, не у всех есть возможность посещать другую страну с целью изучения языка. Погружение в культуру можно производить и при помощи множества медиаматериалов, включающего в себя песни, книги, статьи и фильмы.

Среди препятствий на пути к изучению языка можно выделить отсутствие мотивации, четких целей, непонимание культуры и нехватка дисциплины [6]. Просмотр фильмов и сериалов уже является частью жизни многих людей, а параллельное изучение языка, помимо очевидных плюсов, позволит также воспринимать кино в оригинальном виде. Таким образом, появляется возможность изучать язык в естественной среде с культурным контекстом.

В данной работе рассматривается способ изучения иностранных языков во время просмотра фильмов и сериалов и возможность повышения эффективности изучения при помощи ПО.

1. Цель

Цель курсовой работы состоит в создании улучшенного метода изучения иностранных языков во время просмотра фильмов. Для достижения цели поставлены задачи:

1. Сделать обзор существующих решений.
2. Провести опрос потенциальных пользователей.
3. Реализовать прототип приложения.
4. Протестировать на пользователях.
5. Сделать выводы.

2. Обзор

2.1. Опрос

В процессе исследования появляются вопросы: как люди изучают иностранные языки, как они смотрят кино и насколько заинтересованы в изучении иностранных языков при просмотре кино? Для исследования интересов потенциальных пользователей и получения ответов на указанные выше вопросы был проведен опрос. В опросе участвовали 180 человек, 92% опрошенных возрастом от 16 до 24, 5% от 24 до 29, остальные 3% младше 16.

Изучено было, как часто люди смотрят фильмы и сериалы. 20% опрошенных смотрят кино каждый день, 33% людей — 2-3 раза в неделю, 27% лишь раз в неделю, остальные 20% людей лишь раз в месяц или реже. 40% смотрят кино в одиночку, 45% смотрят в половине случаев с компанией и 15% всегда в компании.

Процесс просмотра у людей заметно различается. 80% людей смотрят фильмы онлайн, у 60% из них нет предпочтения относительно определенного онлайн-кинотеатра. 30% в основном смотрят на мобильных устройствах, остальные — на компьютере. У половины людей телефон под рукой, а 20% людей ставят компьютер подальше при просмотре.

Три самых популярных способа погружать себя в естественную среду языка среди людей — просмотр сериалов и фильмов (60%), чтение книг и статей (55%) и прослушивание песен (63%). Также часть опрошенных отмечают разговор с носителями (30%) и прослушивание аудиокниг (30%).

Обнаружено, что у людей значительно лучше получается читать, чем воспринимать речь на слух и разговаривать. Половина людей отмечает, что часто смотрят фильмы в дубляже, 40% смотрят фильмы в оригинале с русскими или английскими субтитрами, 10% смотрят в основном в оригинале.

В целом люди показали высокую заинтересованность в реализуемом приложении. 50% ответили, что хотели бы учить язык параллельно с

просмотром кино. На просьбу оценить от 1 до 5 заинтересованность в изучении иностранных языков с просмотром фильмов и сериалов, 40% оценило идею на 5, 33% — на 4, 18% на 3 и оставшиеся 9% — на 2 или 1. Из 180 человек 60 человек, то есть треть, оставили почту для получения первых версий приложения.

2.2. Гипотеза

Была сформулирована следующая педагогическая гипотеза. Можно повысить эффективность изучения иностранных языков путем сохранения полного видео- и аудио- контекста видео-отрывками из фильма, а также получением мгновенного перевода целой фразы во время просмотра фильма.

2.3. Существующие решения

Очень мало сервисов, предоставляющих возможность изучать иностранный язык во время просмотра каких-либо видеоматериалов. В российском сегменте существует *LinguaLeo*[10] — 40% опрошенных сообщили, что пользуются этим сервисом для изучения языков. Также есть российский сервис *ororo.tv*[15]. Среди зарубежных решений существуют *fleex.tv*[14] и *FluentU*[4]. У всех почти одинаковая функциональность — показывается дорожка оригинальных субтитров, можно нажимать на любое слово для получения и сохранения его перевода.

2.3.1. LinguaLeo

Самый популярный сервис по изучению языков в России позволяет выбирать из каталога видео, которые можно посмотреть с субтитрами. Там же есть описанная выше функциональность — в любой момент можно остановить и получить перевод выбранного слова или словосочетания. Перевод затем добавляется в словарь пользователя LinguaLeo. Здесь не показывается перевод всей фразы целиком, а видео можно выбирать только из ограниченного набора, фильмов и сериалов в нем

не содержится. Единственный из сервисов работает только с англо-русским переводом.

2.3.2. Ororo.tv

Сайт предоставляет просмотр в описанном выше режиме 56 YouTube-каналов и 61 сериалов. У многих видео и сериалов доступны разные языки субтитров, можно запускать два трека субтитров одновременно.

2.3.3. Fleex.tv

Позволяет смотреть видео на *YouTube / Netflix / TED* или же свои фильмы при помощи скачанного плеера. На *Netflix* просмотр фильмов и сериалов осуществляется при помощи расширения для браузера. Также сервис позволяет изучить слова или выражения до фильма и повторить уже во время просмотра фильма.

2.3.4. FluentU

Интерфейс изучения слов такой же как и у предыдущих, единственное отличие — по умолчанию идут два трека субтитров сразу.

3. Реализация

3.1. Проектирование

Все представленные сервисы позволяют понимать человеку содержание и заучивать переводы данных слов. Знать переводы слов — важная, но не ключевая часть знания языка. В конечном счете требуется умение переводить целые предложения, понимать речь человека и хорошее произношение. Сохранение контекста также помогает запоминать слова и выражения и правильно их применять.

Предлагается вместо сохранения только слов из фразы сохранять отрывки из фильмов с конкретной фразой, так называемые видеокарточки. Во-первых, эмоциональная привязка к моменту в фильме поможет лучше запомнить употребление слова [12]. Во-вторых, полностью сохраняется контекст — сама фраза целиком вместе с аудио- и видеоконтентом. В-третьих, при повторении можно практиковать и перевод, и распознавание речи, и произношение. При показе сперва фразы на одном из языков, пользователь может попробовать перевести и посмотреть весь видеофрагмент. Если показать сначала видеофрагмент, пользователь может попытаться распознать сказанную фразу, а затем попробовать повторить её за персонажем, отрабатывая произношение.

Возможность изучать новые фразы перед просмотром — перспективное направление. Можно отобрать фразы, которые наиболее вероятно не знакомы пользователю, учитывая его словарный запас и уровень языка. Это может быть темой дальнейших исследований и разработки в будущем.

3.2. Технологии

Для реализации требовалось выбрать набор технологий, предоставляющий достаточный контроль над воспроизведением видео, а также возможность создавать свой графический интерфейс для изучения языка после просмотра фильмов, повторять сохраненные ранее моменты и слова. По этой причине плагина для *VLC* [13] было бы недостаточно.

Выбрано такое сочетание технологий, которое позволяет и воспользоваться мощными возможностями плеера *VLC* с программным интерфейсом, предоставляющим широкие возможности управления, и встроить это в общий GUI всего приложения. Использовались *JavaFX* [11], *TornadoFX* [3] для создания GUI и vlcj [9] для получения доступа к проигрывателю *VLC*. Библиотека *vlcj*, написанная на Java[5], предоставляет доступ к библиотеке libvlc на C. При помощи *vlcj* есть возможность устанавливать дорожки субтитров и аудио, ставить на паузу, перематывать время и др. Фреймворк *TornadoFX* представляет собой обертку над библиотекой для создания графических интерфейсов *JavaFX*, используя все возможности по созданию DSL (Domain-Specific Language — предметно-ориентированный язык) на языке *Kotlin*. Для извлечения субтитров из фильма используется командная утилита *mkvtoolnix*[2], хотя в будущем стоит делать это при помощи парсинга *.mkv* файлов в *Java / Kotlin*[7], чтобы убрать нежелательные зависимости от внешних утилит.

3.3. Функциональность

Сперва пользователь выбирает файл / папку с фильмом. Если фильм имеет расширение *.mkv*, при помощи инструмента в командной строке mkvtoolnix происходит извлечение субтитров из *.mkv* файла, если же файл *.avi*, то в папке с фильмом лежат все дорожки субтитров. Затем все дорожки субтитров парсятся при помощи *regexp*-ов и создаются субтитры, который хранит все фразы в дорожке субтитров с их временными метками. Таким образом, для фильма собрана вся нужная информация — путь к самому файлу фильма, все загруженные дорожки субтитров, а также информация о дорожках аудио. Следующим шагом пользователь переходит непосредственно к просмотру фильма, для этого вся информация о фильме передается видеоплееру. Есть возможность открыть окно с выбором дорожек субтитров и аудио, а также задержками у дорожек субтитров. Одна дорожка субтитров будет воспроизводиться на протяжении всего времени, в нижней части экрана,

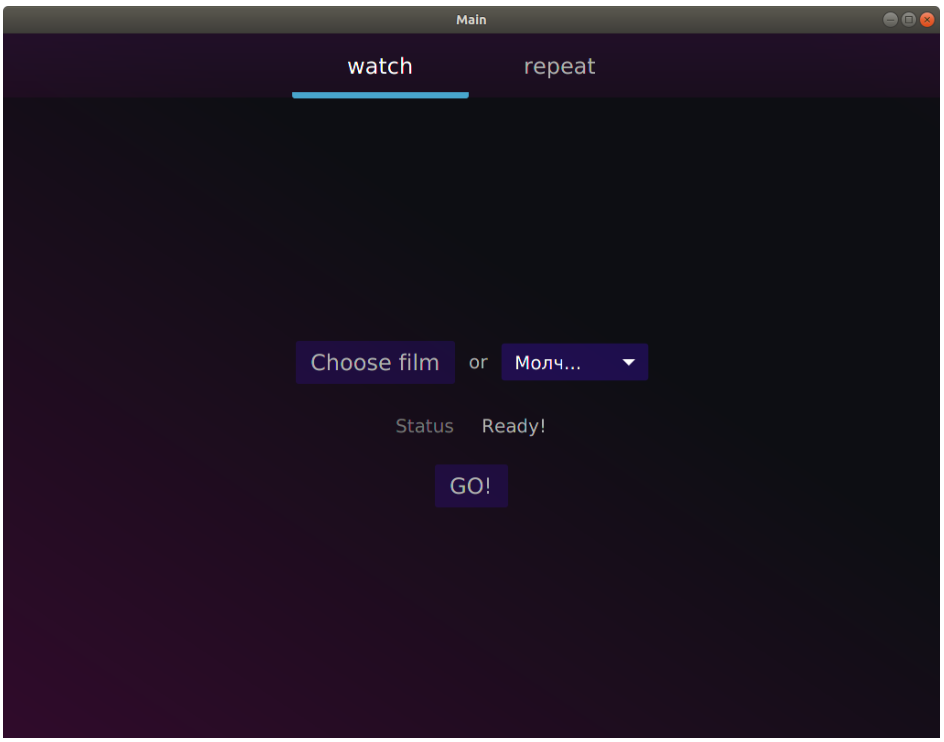

Рис. 1: Начальный экран с возможностью выбора нового или ранее открытого фильма, сверху кнопка для режима повторения предыдущих карточек

а вторая дорожка субтитров будет показываться во время паузы, как показано на рис. 2 и 3. Таким образом, пользователь может, например, выбрать одну дорожку в качестве текста на оригинальном языке, а сверху перевод, таким образом, в моменты, когда понимание теряется, пользователь может остановить и сразу же увидеть перевод. Другой пример — пользователь не ставит дорожку субтитров на нижнюю часть экрана, а для верхней части экрана выбирает оригинальную дорожку субтитров — тогда человек сможет тренировать распознавание речи на слух. Во время паузы также выезжает панель с элементом прокрутки (слайдером) регулировки времени фильма, чтобы перематывать на любой момент, а также несколько кнопок: вернуться назад, поставить на паузу / возобновить воспроизведение, открыть настройки дорожек субтитров и аудио и сохранить отрывок видео — так называемую видеокарточку.

Помимо просмотра фильма можно также открыть режим повторения видеокарточек — видео-отрывков из фильмов с сохраненными суб-

Рис. 2: Фильм проигрывается, снизу отображается только дорожка субтитров на оригинальном языке

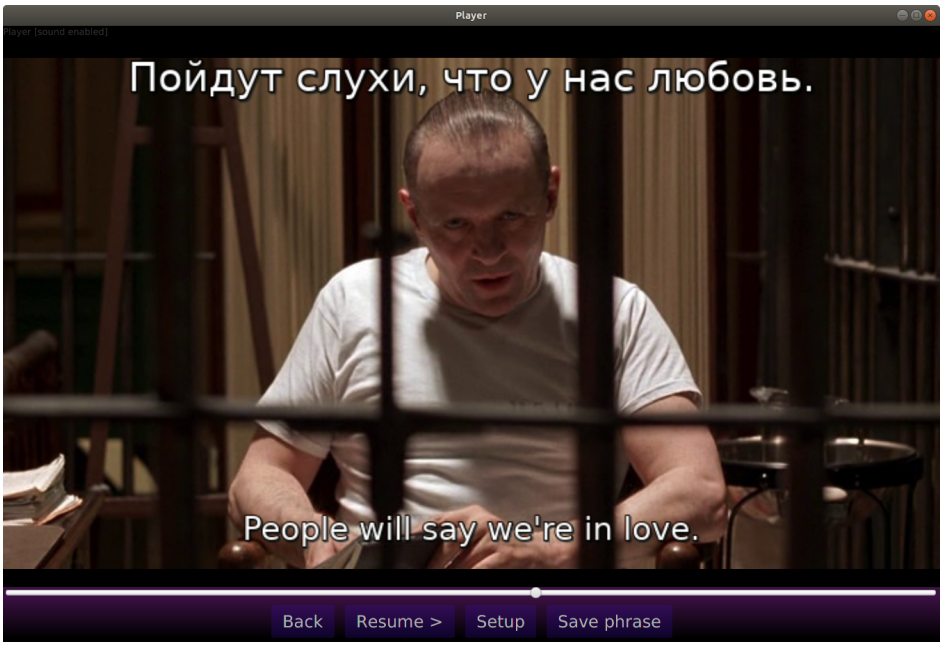

Рис. 3: Фильм поставлен на паузу, сверху отображена фраза из дорожки субтитров на втором языке, в данном случае на русском.

титрами. Есть два варианта: сохранять видеокарточки в виде отрывков видео, отдельных .mkv файлов с вырезанными соответствующими субтитрами, и сохранять лишь временные метки, путь к фильму и номера выбранных дорожек субтитров и аудио. На момент тестирования со-

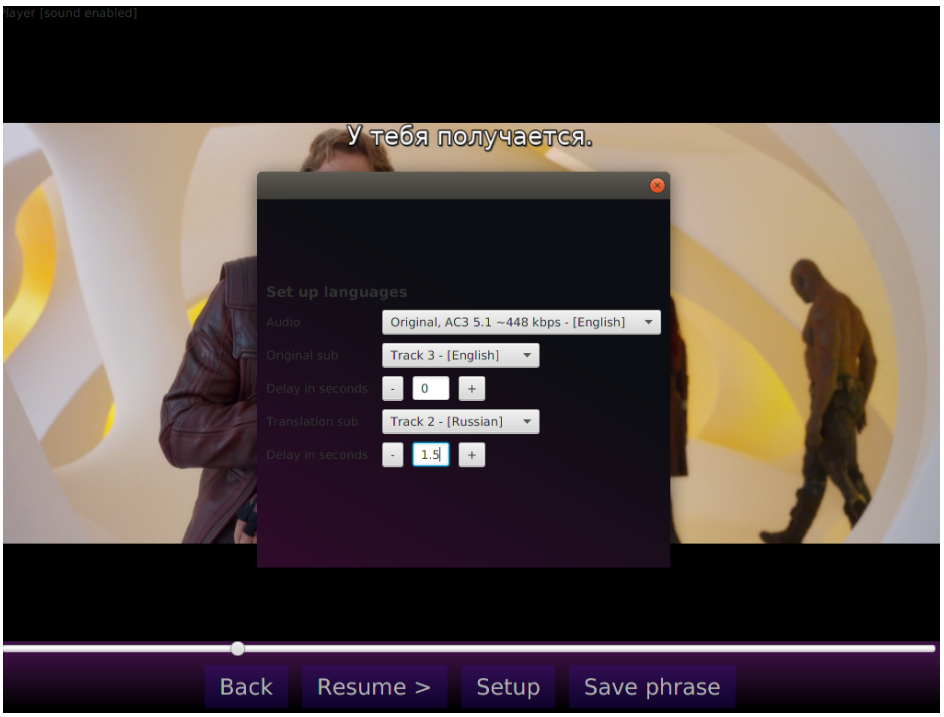

Рис. 4: Окно выбора дорожек субтитров и аудио, а также задержек у дорожек субтитров

хранение *.mkv* файлов при помощи *vlcj* работало очень нестабильно, поэтому выбран второй вариант. Для прототипа сохранение происходит путем дописывания строчки со всей указанной выше информацией в текстовый файл. Когда пользователь выбирает режим повторения видеокарточек, текстовый файл парсится и для текущего фильма создается список из *VideoMoment*, каждый из которых хранит временные метки и номера выбранных дорожек для определенной видеокарточки.

Была проверена гипотеза о полезности изучения слов перед фильмом. Для этого был написан скрипт на *Python* с использованием библиотеки для обработки естественного языка *nltk*, который извлекает все слова из субтитров, фильтрует имена собственные и часто встречающиеся слова — междометия, союзы, артикли, а также приводит все слова к единообразному виду — ставит в одно склонение или приводит к одной части речи похожие слова. Обработанная дорожка субтитров загружается в основной программе и слова сортируются от наиболее часто встречаемых к наиболее редко встречающимся, из списка убираются слова, которые уже присутствуют в словаре пользователя. Однако на

практике обнаружились некоторые недостатки у изучения слов перед просмотром фильмов. Слова без контекста не очень хорошо запоминаются и усваиваются. Если же вводить контекст в виде предложения, в котором это слово используется, то пользователь может увидеть так называемые спойлеры — преждевременно раскрытую важную сюжетную информацию. Также часто пользователь хочет приступить непосредственно к просмотру фильма, без предварительного изучения слов. По этим причинам было принято решение отказаться от такого подхода.

3.4. Архитектура

В фреймворке *TornadoFX* есть естественные инструменты для реализации паттерна *Model-View-Controller (MVC)*, а именно классы типов *View* и *Controller*. При наследовании этих классов создаются *Singleton*-ы, т.е. для каждого наследника класса View существует только один экземпляр этого класса. Все графические переходы осуществляются от одного *View* к другому, ключевая логика не хранится в классах *View*. Стоит отметить, что большая часть взаимодействий и логики работает на объектах *Property*: у них можно определить действия, которые происходят при изменении объекта, что является паттерном *Observer*. Таким образом можно избегать многочисленных связей между объектами и частями кода. Например, *isPausedProperty* определяется как объект *SimpleBooleanProperty*, поэтому все действия, которые происходят на паузе или продолжении просмотра, независимо привязываются при помощи метода, принимающего функцию *isPausedProperty.onChange{...}*.

Создан класс *SubTitle*, предназначенный для хранения всех фраз из дорожки субтитров с их временными метками для последующего нахождения нужной фразы по временной метке при помощи бинарного поиска в упорядоченном списке фраз. Фразы из дорожки субтитров представлены в классе *SubPhrase*, время представлено в отдельном классе *Time*.

Работа приложения начинается с класса *FilmPlayerApp*, который сразу же переходит в *MainView* (рис. 1). В *MainView* после выбора пути

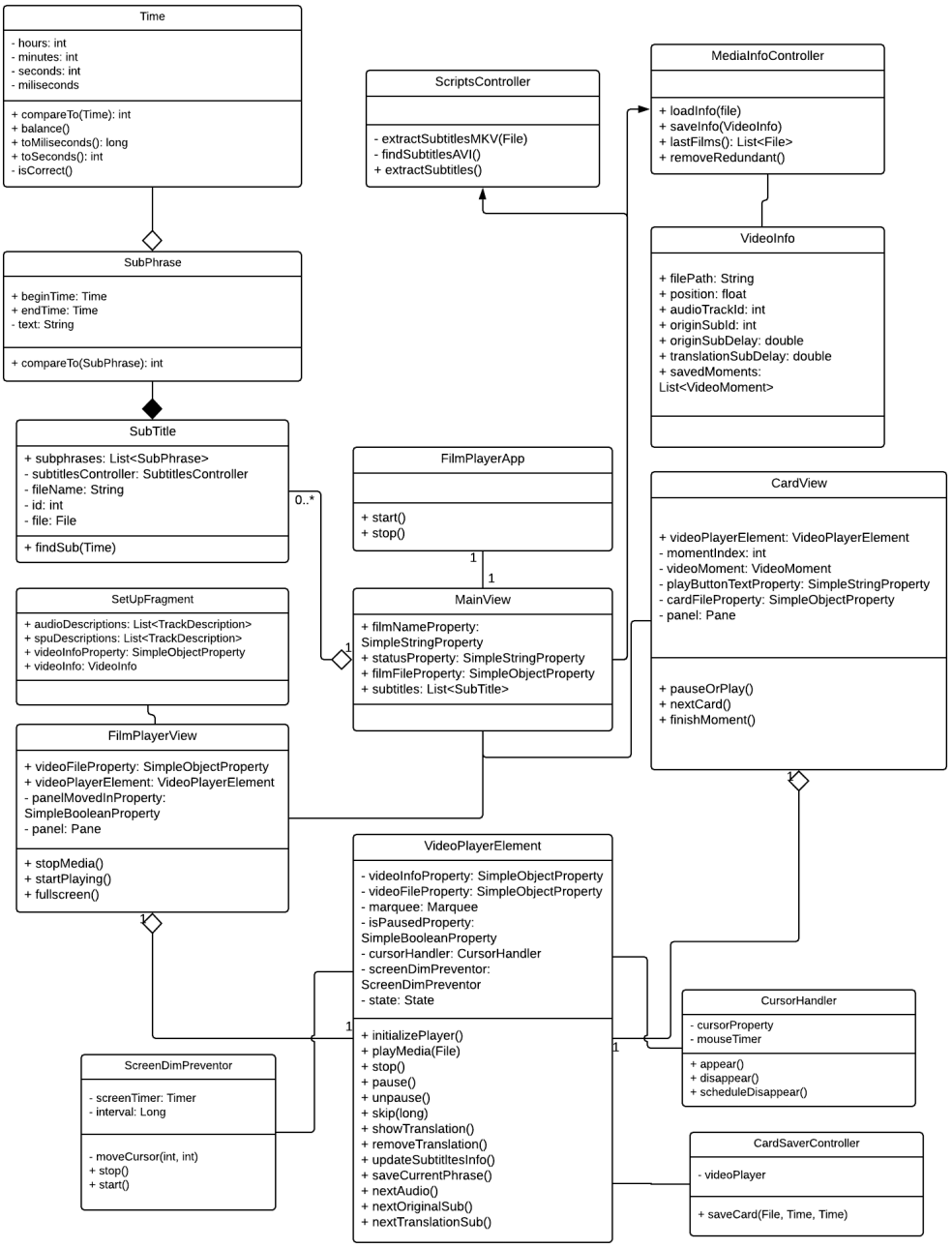

Рис. 5: UML-диаграмма классов

к фильму происходит вызов *ScriptsController* для извлечения и загрузки субтитров, которые сохраняются в поле subtitles класса *MainView*. Затем в зависимости от того, решил пользователь смотреть фильм или повторять карточки, идет переход в *FilmPlayerView* или *CardView*. Оба *View* используют ключевой класс *VideoPlayerElement* в качестве видеоплеера.

Класс *VideoPlayerElement* использует класс *VideoPlayer*, предостав-

ляющий доступ к видеоплееру на *vlcj*. В классе *VideoPlayerElement* реализованы методы для переключения дорожек субтитров и аудио, обработки и запуска плеера *VideoPlayer*, а также для нахождения нужной строчки из дорожки субтитров для показывания сверху во время паузы. Помимо этого *VideoPlayerElement* предотвращает погасание экрана во время просмотра фильма, используя работающий на *Timer* класс *ScreenDimPreventor*, который незаметно передвигает курсор через равные промежутки времени. Во время просмотра также исчезает курсор, что достигается благодаря классу *CursorHandler* — он меняет иконку курсора на пустую через некоторое время после продолжения просмотра.

Класс для просмотра фильмов *FilmPlayerView* (рис. 3) помимо класса *VideoPlayerElement* создает также объект класса *Pane* — нижнюю панель, которая присвоится в *VideoPlayerElement* и будет показываться во время паузы. Панель включает в себя элемент прокрутки (слайдер) и четыре кнопки, одна из которых ведет обратно на *MainView*, другая открывает *SetUpFragment* — окно для выбора дорожек субтитров и аудио. Когда просмотр фильма заканчивается: закрывается программа или происходит возврат на *MainView*, класс *VideoInfo* с информацией о фильме, выбранных дорожках субтитров, списке сохраняемых видеокарточках VideoMoment и последней временной метке сохраняется. Запись происходит в текстовый файл для прототипа, в будущем просто заменить на обращение к базе данных. Повторение видеокарточек происходит в окне класса CardView, куда передается *VideoInfo*, хранящий в себе список *VideoMoment*, а также инициализируется новый *VideoPlayerElement*. Как и в *FilmPlayerView*, создается панель с элементом прокрутки (слайдером) и двумя кнопками — вернуться назад на *MainView* и перейти к следующей видеокарточке.

3.5. Ограничения

У приложения существуют некоторые ограничения в переносимости и использовании, которые в будущем стоит исправить. Во-первых, за-

висимость от утилиты в командной строке *mkvtoolnix* стоит заменить кодом на *Java / Kotlin*. Во-вторых, программа протестирована пока лишь на различных версиях *Ubuntu*, на *OS X* в будущем стоит устранить некоторые проблемы с зависимостями. В-третьих, видеокарточки сохраняются не в виде отрывков фильма, а в виде набора временных меток начала и конца отрывка и информации о выбранных дорожках субтитров и аудио, поэтому воспроизведение видеокарточек возможно только для определенного выбранного фильма. В-четвертых, поддерживаются только *.mkv* и *.avi* файлы, впрочем, большинство фильмов представлено в одном из этих двух форматов, поэтому это не кажется большой проблемой.

4. Анализ отзывов пользователей

По получении готового прототипа Desktop-приложения 10 пользователям была дана возможность пользоваться приложением, чтобы получить отзывы для подтверждения или опровержения гипотезы. В качестве примеров будут приведены два отзыва.

“В целом приложение оставляет позитивное впечатление, в первую очередь за счет гибких настроек интерфейса просмотра фильма. Многие платформы для изучения языков имеют подобный инструмент, но интерфейс всегда одинаков, и зачастую требует больше действий для автоматического перевода фразы, в то время как в данном приложении я могу настроить удобный мне способ показа и задержи дорожек с субтитрами. Визуальное оформление выглядит несколько сырым, однако на функциональности это не сказывается никак. Особо хотелось бы отметить функцию добавления в картотеку сложных моментов с возможностью их последовательного повтора и изучения”

“После пользования приложением в течение месяца остался только положительный опыт. Новый подход к сочетанию просмотра фильмов и изучению иностранного языка позволяет объединить полезное с приятным, моментальный доступ к переводу слову не отрывает от просмотра фильма, а сохранение отдельных отрывков дает возможность тренировать потом сложные для понимания участки. Отдельно стоить отметить возможность управления курсором, не надо тянуться к клавиатуре, чтобы поставить на паузу. Сам видеоплеер работает стабильно, фильм не зависает, удобные настройки для выбора основных и дополнительных субтитров. Было бы интересно увидеть интеграцию с сервисами по изучению языков.”

Во всех отзывах прослеживается отмечаемая польза от приложения при изучении языков без затруднения просмотра фильма. Также пользователи отмечают, что особенно полезно изучать видеокарточки после просмотра фильмов, а благодаря всему видео-отрывку это запоминается гораздо лучше и надежнее. Таким образом, это является свидетельством в пользу изначальной гипотезы.

Заключение

В рамках работы над курсовой в осеннем семестре были выполнены следующие задачи:

1. Рассмотрены существующие решения.
2. Опрошено 180 человек для понимания нужд пользователей.
3. Начата работа над прототипом.
4. Закончить рабочий прототип.
5. Протестировать на пользователях.

Список литературы

[1] Beatty Ken. Teaching and Researching Computer-Assisted Language Learning.

[2] Bunkus Moritz. MKVToolNix: a set of tools to create, alter and inspect Matroska files. — 2019. — URL: https://mkvtoolnix.download/ (дата обращения: 12.05.2019).

[3] Edvin. TornadoFX: JavaFX Framework for Kotlin. — 2019. — URL: https://github.com/edvin/tornadofx/ (дата обращения: 12.05.2019).

[4] FluentU. — 2019. — URL: https://www.fluentu.com (дата обращения: 12.05.2019).

[5] Java. Programming language. — 2019. — URL: https://www.java.com/ru/about/whatis_java.jsp (дата обращения: 12.05.2019).

[6] Khan Intakhab, Weeber Stan. Barriers in the Learning of English: An Exploratory Study // British Journal of Education, Society Behavioural Science. — 2016. — 03. — Т. 15.

[7] Kotlin. Programming language by JetBrains. — 2019. — URL: https://kotlinlang.org/ (дата обращения: 12.05.2019).

[8] Language therapy and bilingual aphasia: Clinical implications of psycholinguistic and neuroimaging research / Ana Inés Ansaldo, Karine Marcotte, Lilian Scherer, Gaelle Raboyeau // Journal of Neurolinguistics. — 2008. — Т. 21, № 6. — С. 539 – 557. — URL: http://www.sciencedirect.com/science/article/pii/S0911604408000146.

[9] Limited Caprica Software. vlcj: Java framework for the VLC media player. — 2019. — URL: https://github.com/caprica/vlcj (дата обращения: 12.05.2019).

[10] LinguaLeo. — 2019. — URL: https://lingualeo.com (дата обращения: 12.05.2019).

[11] Oracle. JavaFX. — 2019. — URL: https://wiki.openjdk.java.net/display/OpenJFX/Main (дата обращения: 12.05.2019).

[12] Synaptic evidence for the efficacy of spaced learning / Enikö A. Kramár, Alex H. Babayan, Cristin F. Gavin и др. // Proceedings of the National Academy of Sciences. — 2012. — Т. 109, № 13. — С. 5121–5126. — https://www.pnas.org/content/109/13/5121.full.pdf.

[13] VLC. — 2019. — URL: https://www.videolan.org/vlc/index.html (дата обращения: 12.05.2019).

[14] fleex.tv. — 2019. — URL: https://fleex.tv (дата обращения: 12.05.2019).

[15] ororo.tv. — 2019. — URL: https://ororo.tv (дата обращения: 12.05.2019).