



Универзитет у Београду
Електротехнички факултет

ДИПЛОМСКИ РАД

**Имплементација WebVTT парсера
за титлове на видео записима
на језику C++**

Ментор:

проф. др Драган Милићев

Студент:

Душан Стијовић 2017/0145

Београд
септембар 2021.

Садржај

1. Увод.....	1
2. Опис проблема	3
2.1. Поређење са другим форматима.....	3
2.1.1. <i>SRT</i>	3
2.1.2. <i>TTML</i>	4
2.2. Употреба <i>WebVTT</i> формата	4
2.2.1. <i>MPEG-DASH</i>	4
3. <i>WebVTT</i> формат	7
3.1. Увод.....	7
3.2. Блок са заглављем	7
3.3. Блок са преводом видеа	8
3.3.1. Идентификатор.....	8
3.3.2. Временска ознака	8
3.3.3. Додатна подешавања	9
3.3.4. Текст	12
3.4. Регион.....	15
3.4.1. Идентификатор (<i>енгл. id</i>).....	15
3.4.2. Ширина (<i>енгл. width</i>).....	15
3.4.3. Број линија (<i>енгл. lines</i>).....	15
3.4.4. Начин додавања новог блока са преводом	15
3.4.5. Тачка која представља регион (<i>енгл. region anchor</i>).....	16
3.4.6. Позиција региона на екрану (<i>енгл. view port anchor</i>).....	16
3.5. Стили (<i>енгл. stylesheets</i>)	16
3.5.1. Стили за регионе (<i>енгл. ::cue-region</i>).....	16
3.5.2. Стили за блокове са текстом (<i>енгл. ::cue</i>)	17
4. Имплементација	20
4.1. Бафер за стринг	20
4.2. Главни део парсера	20
4.3. Парсирање блока са текстом.....	22
4.4. Парсирање блока са стилем	24
4.5. Примена одговарајућег стила на одговарајући блок	25
4.6. Подршка за обилазак стабла превода.....	26
5. Корисничко упутство.....	28

5.1. Бафер за стринг	28
5.2. Бафер за блокове	28
5.3. Декодер	29
5.4. Парсер	30
5.5. Обилазак стабла	30
6. Додавање библиотеке у постојећи пројекат	33
6.1. Клонирање пројекта са <i>GitHub-a</i>	33
6.2. Прављење дељење библиотеке	34
6.3. Покретање примера коришћења	35
6.4. Додавање библиотеке приликом превођења	35
6.5. Покретање програма који је повезан са дељеном библиотеком.....	36
6.6. Ограничења и препоруке.....	37
7. Закључак	39
8. Литература.....	40
Списак слика.....	1
Списак табела	2

1. Увод

Циљ овог рада је имплементација парсера за *WebVTT* [1] (енгл. *Web Video Text Tracks*), чија је намена да се користи приликом приказа видео записа. *WebVTT* формат се користи као стандард за представљање текста који служи као превод уз видео записе (тзв. титла). Уз то, овај формат омогућава разнолико стилизовање и позиционирање тог текста. Занимљиво је да сам текст који се преноси нема специфичан формат, тако да се може преносити произвољан формат текста. На овај начин *WebVTT* формат се може користити и за неке друге потребе осим за видео записе. [2] На пример, може се преносити текст превода у *JSON* [3] формату. Како су видео снимци постали веома распрострањени на вебу, а како је свака платформа за приступ записима имплементирала своје протоколе и свој формат фајла са садржајем (а самим тим и формат фајла за превод), јавила се потреба за стандардом за титлове видео записа. Овим је омогућено да више уређаја и платформи буду међусобно подржане. Одлучено је да се не користе већ постојећи формати, јер су или подржавали све што је било потребно, а били превише комплексни, или су им недостајале потребне функционалности. Специфични системи који су били циљеви *WebVTT* формата су *HTML-5* [4] системи. Из тог разлога *WebVTT* формат поседује добру подршку за *HTML-5* и може се релативно лако интегрисати са њим. Скоро сви веб претраживачи имплементирају *WebVTT* формат [5]. Како је стандард формата релативно нов, потпуну подршку формата подржава свега неколико претраживача. Наравно, формат се може користити и без *HTML-5*, као што је случај код Андроида.

На почетку овог рада извршено је поређење *WebVTT* формата са неким од других представника формата текстова за преводе на видео записима. Такође, дат је пример примене формата код *MPEG-DASH* [6] протокола. Објашњена је структура протокола, као и то на који начин је могуће преносити текст са преводом видеа уз видео снимак.

Следећи део рада садржи опис самог формата. Објашњена је структура фајла, које све опције поседује формат, које је њихово значење и на који начин корисник може да их користи.

Након излагања о структури формата прелази се на главни део рада, имплементацију парсера. Помоћу упрошћених дијаграма класа објашњена је структура најзначајнијих делова система. Неки од делова система који су објашњени су бафер стрингова, декодер и парсер. Бафер стрингова служи за дохватање стринга који се парсира. Посебан тип бафера је потребан због могућности да није доступан цео фајл, него да пристиже део по део фајла. Пошто је формат *UTF-8* енкодован, потребно га је трансформисати у формат који је лакши за обраду, па је потребан декодер. Парсер је потребан како би се фајл обрадио и како би се направили одговарајући елементи формата који су потребни за приказ текста.

У посебном поглављу објашњен је начин коришћења овог парсера. Помоћу дијаграма секвенци објашњено је неколико сценарија његовог коришћења.

Посебан део поглавља о коришћењу система се бави повезивањем са другим системом. Како би корисник могао да користи имплементиране функционалности, потребно је повезати два система. У поглављу је објашњено како је могуће скинути изворни код, превести га, повезати са неким другим системом и како покренути извршни фајл повезаног система. Пошто је систем развијан под одређеним условима, под тим условима се може и користити, па су из тог разлога дате препоруке и ограничења коришћења система.

На самом крају, у закључку, дат је осврт на изложен проблем, његово решење и дати су предлози за унапређење система.

2. Опис проблема

Како се у данашње време све више времена проводи користећи паметне уређаје и гледајући видео снимке на многим светским језицима, постоји потреба за форматом титлова за видео снимке који ће подржати све потребе. Релативно нов формат за видео титлове је *WebVTT*, чији стандард је тренутно у процесу прихватања [2]. Формат подржава све што би се могло тражити од видео титлова, док задржава једноставност и врло лако се може интегрисати са *HTML-ом*, који се најчешће користи за репродукцију видео садржаја. Пре увођења овог формата постајали су други формати, али су или били превише једноставни, јер нису подржавали све потребне функционалности, или су били превише компликовани за интегрисање. *WebVTT* формат, осим основне функционалности за дефинисање појављивања титлова у одређеним временским интервалима, подржава и разна подешавања самих титлова. Он омогућава подешавање тога где ће се титлови наћи на екрану, да ли ће се приказивати хоризонтално или вертикално, њихову величину, поравнање, као и многе друге опције. Осим подешавања самих титлова, могуће је њихово груписање у регионе и позиционирање региона на одређени део екрана. Такође, он подржава разноврсно стилизовање преко *CSS-а* [7], што је врло корисно уколико неке ствари треба истаћи како би особе које имају потешкоће могле да га лакше прочитају. Осим подршке за титлове, формат се може користити и за пренос разних метаподатака. Сам садржај титлова нема специфичан формат, тако да се на тај начин могу послати разне информације. Битна ствар је да се формат преноси помоћу *UTF-8* [8] енковања, тако да подржава све језике, а осим тога, омогућено је да се приликом имплементације парсера одлучи које ће се функционалности подржати, па је тако на пример могуће имплементирати парсер који нема подршку за *CSS*.

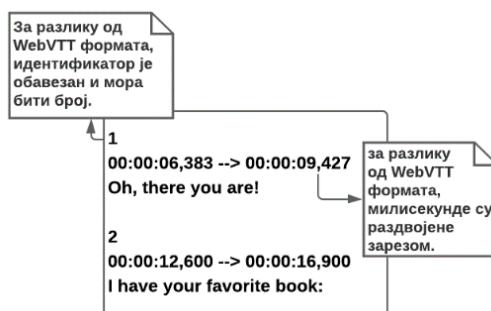
Осим *HTML-а*, видео садржај се репродукује и на Андроиду. Из тог разлога јавља се потреба за парсером који није директно везан са *HTML-ом*, као што је то случај са *JavaScript-ом*. Већина парсера за формат *WebVTT* је управо писана на JavaScript-у. Из тог разлога ће у наставку бити објашњена имплементација *WebVTT* парсера.

2.1. Поређење са другим форматима

У овом поглављу ће укратко бити објашњени неки формати који се такође користе за видео титлове, као и разлике у односу на *WebVTT* формат.

2.1.1. *SRT*

SRT (SubRip Text) представља једноставан формат. Састоји се само од блокова текста. Сваки блок са текстом се састоји из идентификатора који је број, временске ознаке почетка и краја појаве титла и самог текста титла. Овај тип формата не подржава позиционирање и стилизовање. Због своје једноставности и недостатка могућности за форматирање, овај формат није много заступљен. На следећој слици се налази пример фајла у датом формату. [9]



Слика 1 Пример фајла у SRT формату

2.1.2. TTML

TTML (Timed Text Markup Language) представља комплексан формат. Формат се пише у *XML*-у [10], а синтакса му је слична *HTML*-у. Екстензија фајла је „.ttml“. Формат је дизајниран са сврхом да обједини све функционалности постојећих формата и да тако постане стандард за размену између апликација. [11] Формат претежно користи индустрија за репродукцију видео снимака, док се *WebVTT* формат користи претежно у веб претраживачима. На следећој слици се налази пример коришћења формата.

```
<tt xml:lang="en" xmlns="http://www.w3.org/ns/ttml" xmlns:tts="http://www.w3.org/ns/ttml#styling">
  <head>
    <layout>
      <region xml:id="rTop" tts:origin="10% 10%" tts:extent="80% 20%" />
      <region xml:id="rMiddle" tts:origin="10% 40%" tts:extent="80% 20%" />
      <region xml:id="rBottom" tts:origin="10% 70%" tts:extent="80% 20%" />
    </layout>
  </head>
  <body>
    <div xml:lang="en">
      <p begin="0.76s" end="3.20s" region="rTop">
        I sent a message to the fish:
      </p>
      <p begin="3.20s" end="6.61s" region="rMiddle">
        I told them "This is what I wish."
      </p>
    </div>
  </body>
</tt>
```

Слика 2 Пример употребе TTML формата

2.2. Употреба WebVTT формата

У овом одељку биће објашњена употреба *WebVTT* формата на примеру протокола *MPEG-DASH*. Биће објашњени основни делови протокола, као и на који начин се преноси превод уз видео снимак.

2.2.1. MPEG-DASH

MPEG-DASH (Dynamic Adaptive Streaming over HTTP) представља протокол за пренос видео фајла преко интернета. Током година је расла потреба за видео снимцима на интернету, док је сваки произвођач правео свој систем са својим протоколима. Због тога се појавила потреба да се направи стандард који ће користити сви произвођачи, а који ће бити компатибилан са свим производима. Сам процес преноса видеа се састоји из следећих делова:

- Сегментација

Видео снимак се прво дели на сегменте, односно на мање видео снимке, од којих сваки траје од 2 до 4 секунде.

- Енковање

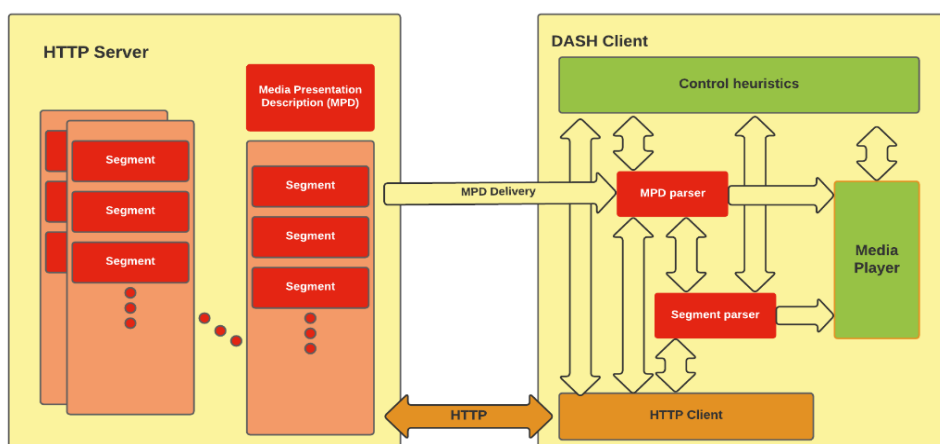
Сваки сегмент који је направљен трансформише се у неки стандардан формат, како би страна која прима сегмент могла да га интерпретира.

- Пренос преко интернета

Када корисник започне гледање видеа, сегменти се шаљу корисниковом уређају путем интернета. Најчешће се користи *CDN* [12] како би се сегменти дистрибуирали ефикасније.

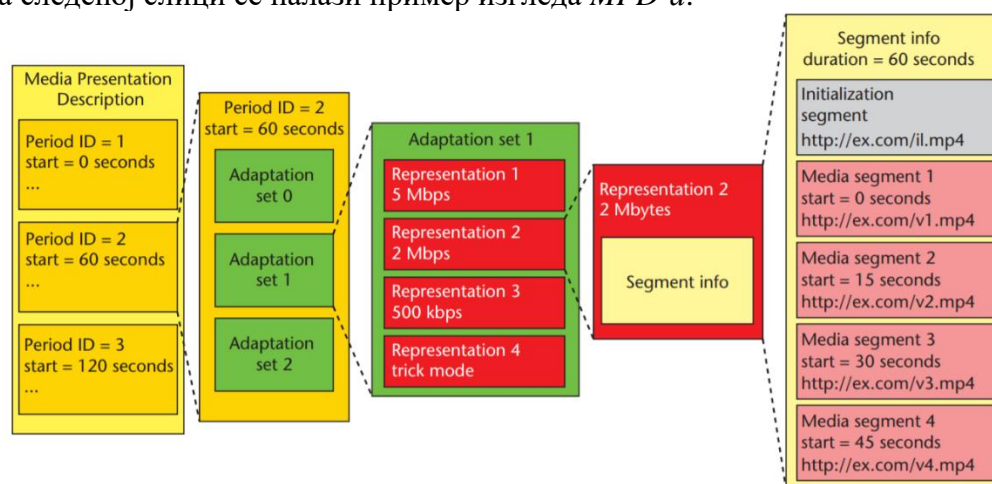
Једна од значајних ствари које поседује *MPEG-DASH* је промењив проток бита (*енгл. ABR Streaming*). Како сегменти видео снимака могу да постоје у више различитих протока бита, односно резолуције, *MPEG-DASH* клијент може да ослушкује мрежу и да квалитет слике прилагођава стању мреже. На тај начин се омогућава да корисник гледа снимак у континуитету независно од стања мреже.

Енковани сегменти се чувају на серверу. Осим самих сегмената, на серверу се налази и тзв. *MPD (Media Presentation Description)*, који описује сегменте. Како би се пустио видео садржај, клијент мора прво да дохвати *MPD*. Након дохватања, клијент парсира *MPD* и тако сазнаје разне информације о видеу - минималну и максималну резолуцију сегмената, која сва енковања су подржана, дигитална права видеа, локацију на интернету и многе друге информације [13]. За дохватање *MPD-a* и сегмената најчешће се користи *HTTP* протокол. Прво је потребно довући одређени део видеа, како би се могло на време реаговати на промене на мрежи. Након тога, видео се пушта и наставља се довлачење следећих сегмената уз праћење стања на мрежи. Уколико дође до промене на мрежи, видео клијент ће реаговати и прилагодити резолуцију видеа новом стању мреже. Спецификацијом је обухваћен *MPD* и формат сегмената. Пренос *MPD-a*, начини енковања сегмената, дохватање *MPD-a* и сегмената, хеуристике за промену резолуције, као и само пуштање видеа не спадају под стандард и остављено је на клијенту да одлучи на који начин ће бити имплементирани. На следећој слици се налази инфраструктура за гледање видеа користећи *MPEG-DASH* протокол. Црвени правоугаоници спадају под стандард, док остали зависе од имплементације клијента [13].



Слика 3 Инфраструктура MPEG-DASH клијента [13]

MPD представља најзначајнији део *MPEG-DASH-a*. У њему је описан сваки део снимка. Сваки део снимка имај свој идентификатор (цео број), почетно време тог дела у видео снимку и скупове за прилагођавање (енгл. *adaptation set*). Сваки скуп за прилагођавање садржи једну или више компонената снимка у разним начинима енкодавања. Компоненте снимка су видео, аудио и превод. На пример, један скуп за прилагођавање може да има различите резолуције видео компоненте истог видеа, док други скуп садржи превод видеа. Сваки скуп за прилагођавање се састоји од више репрезентација. Свака репрезентација представља енкодовану алтернативу исте компоненте видеа. Разликује се од осталих репрезентација по више ствари, од којих је једна резолуција. Репрезентација се састоји од једне или више информације о сегментима. Свака информација о сегменту садржи идентификатор сегмента (цео број), почетно време сегмента у видео снимку, као и линк ка серверу где се налази дати сегмент. На следећој слици се налази пример изгледа *MPD-a*.



Слика 4 Изглед *MPD-a* [13]

Осим наведених основних функционалности, протокол поседује и оне напредне. Једна од њих је подршка за рекламе. Протокол имплементира интерфејс помоћу кога се могу убацити рекламе у оригинални видео снимак. Пошто је снимак подељен на сегменте, лако је убацити нов сегмент, у којем ће бити реклама, између два већ постојећа сегмента. Детаљан опис осталих функционалности прелази оквире овог рада. Препоручује се кориснику да се упозна са спецификацијом протокола уколико га интересују све функционалности које протокол садржи.

3. WebVTT формат

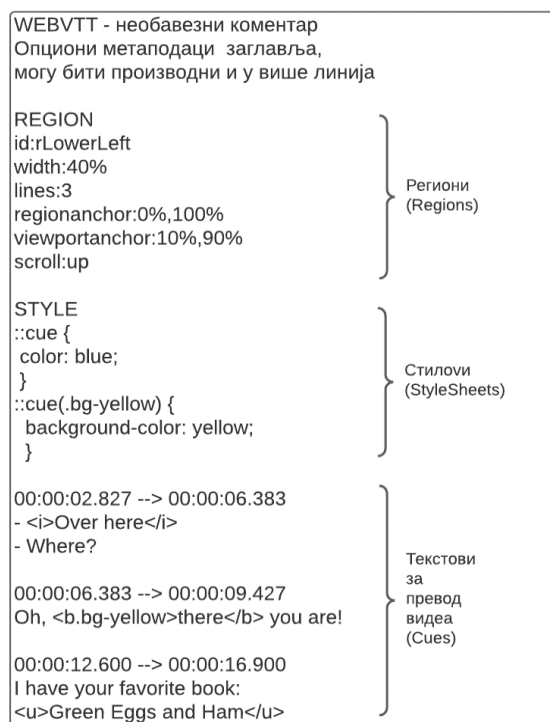
Садржај у *WebVTT* запису треба да буде *UTF-8* енкодован. Осим самог текста и временских ознака почетка и краја одрађеног текста, формат подржава разне могућности за стилизовање и позиционирање текста. Екстензија фајла је „.vtt“. У наставку поглавља ће бити објашњена структура формата, као и опције које формат нуди. [2]

3.1. Увод

WebVTT формат се састоји од следећих блокова:

- Заглавље
- Стил (*енгл. stylesheet*)
- Регион (*енгл. region*)
- Коментар (*енгл. comment*)
- Текст титла (*енгл. cue*)

Обавезни блок је заглавље, док су остали блокови опциони. У наставку ће бити објашњен сваки појединачни блок. На следећој слици се налази пример *WebVTT* формата.



Слика 5 Пример структуре *WebVTT* формата

3.2. Блок са заглављем

Блок за заглављем (*енгл. Header*) се појављује на почетку фајла. Прва линија *WebVTT* фајла мора почети са низом знакова *WEBVTT*, кога прати бели знак¹. У наставку линије може се наћи опциони коментар, а у новом реду метаподаци заглавља. Сам стандард не описује метаподатке

¹ Знак размака, знак табулације или знак за прелаз у нови ред.

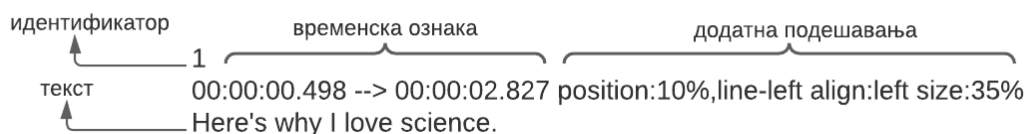
заглавља, тако да они могу бити произвољног формата и значења, а најчешће се налазе информације које детаљније описују садржај фајла. Метаподаци заглавља су опциони. Пример метаподатака у препорученом формату се налази на слици испод [9].

```
WEBVTT
Kind: captions
Language: en-US
Channel: CCl
Station: Online ABC
ProgramID: SH010855880000
ProgramType: TV series
ProgramName: Castle
Title: Law & Murder
Season: 3
Episode: 19
PublishDate: 2011-03-28
ContentAdvisory: TV-14
```

Слика 6 Пример метаподатака заглавља [9]

3.3. Блок са преводом видеа

Блокови са текстом за превод титла (*енгл. Cue*) представљају главни блок *WebVTT* записа и садрже текст који ће ићи као превод онога што је изречено у видеу. После појаве првог блока са преводом видеа, занемарују се појављивања осталих врста блокова. На слици се могу видети основни делови овог блока.



Слика 7 Приказ основних делова блока са преводом

У наставку ће бити објашњен сваки део блока са преводом. Обавезни делови су текст и временска ознака почетка и краја појављивања датог текста, док је идентификатор опцион.

3.3.1. Идентификатор

Идентификатор представља једну линију текста, која јединствено представља блок са преводом. Сваки блок са преводом треба да има јединствен идентификатор. Он се може користити приликом стилизовања блока, како би се одређени стил применио само на блок са датим идентификатором. Може бити једноставан, као број, а може бити и произвољно комплексан. Такође, пошто је *WebVTT* UTF-8 енкодован, може се написати у било ком писму [9].

3.3.2. Временска ознака

Временска ознака се задаје у формату као на следећој слици. Прва временска ознака представља тренутак почетка приказа текста на видеу, а друга временска ознака тренутак када треба престати са приказивањем текста на видеу. Ова два времена су раздвојена низом карактера „-->“². Поље за сате у временској ознаци може бити изостављено, док су остала поља обавезна. У истој линији као временска ознака, прецизније у наставку исте линије, могу се наћи додатна подешавања блока са преводом [9].

² *HYPHEN-MINUS* (U+002D), *HYPHEN-MINUS* (U+002D) и *GREATER-THAN SIGN* (U+003E)

hours:minutes:seconds.milliseconds --> hours:minutes:seconds.milliseconds

Слика 8 Формат временске ознаке

3.3.3. Додатна подешавања

Додатна подешавања се задају после временске ознаке за почетак и крај текста са преводом, одвојена знаком размака или табом. Подешавања се задају по формату име:вредност. Не сме да постоји неки бели знак са леве или десне стране знака „:“ Свако подешавање одвојено је од следећег размаком или табом. У наставку ће бити описана додатна подешавања.

3.3.3.1. Правац текста (енгл. *text direction*)

Ово подешавање се задаје командом `vertical`. Уколико ово подешавање није присутно, подразумевано приказивање текста је хоризонтално, а уколико је подешавање присутно текст ће се приказивати вертикално [9].

Могуће вредности су:

- `lr` – Текст расте слева надесно, односно нова линија текста се додаје са десне стране тренутне линије
- `rl` - Текст расте здесна налево, односно нова линија текста се додаје са леве стране тренутне линије

3.3.3.2. Поравнање текста (енгл. *text alignment*)

Ово подешавање се задаје именом `align`, а вредност може бити нека од следећих: `start`, `left`, `center`, `middle`, `right` или `end` [9]. Ефекат поравњања зависи од вредности подешавања правца текста, и може се видети у следећој табели.

Табела 1 Ефекат поравњања у зависности од правца текста [9]

Подешавање	Правац хоризонталан	Правац вертикалан
<code>align:start</code> <code>align:left</code>	Текст креће са леве стране екрана	Текст креће са горње стране екрана
<code>align:center</code> <code>align:middle</code>	Текст се налази на средини	Текст се налази на средини
<code>align:end</code> <code>align:right</code>	Текст креће са десне стране екрана	Текст креће са доње стране екрана

3.3.3.3. Позиција (енгл. *position*)

Ово подешавање се задаје командом `position`. Ефекат подешавања зависи од вредности подешавања правца текста и од вредности подешавања поравњања текста. Ефекат поравњања на позицију је да се специфицира који део текста се позиционира (леви, десни или центар текста), аналогно са датим вредностима за подешавање поравњања. Вредност подешавања се даје у процентима од 0 до 100 и треба да садржи и знак проценат. Утицај подешавања поравњања текста се може надјачати експлицитним навођењем `line-left`, `center` или `line-right`

запетом одвојеним од вредности процента позиције. Овим се специфицира који део текста се позиционира (леви, десни или центар текста) [9]. Ефекат подешавања дат је у следећој табели.

Табела 2 Ефекат поравњања у зависности од правца текста, поравњања текста и позиције [9]

Подешавање	Хоризонталан правац текста	Вертикалан правац текста
position:0% align:start	Лева страна текста је на левој ивици екрана.	Горња ивица текста је на горњој ивици екрана.
position:0% align:center	Центар текста је на левој ивици екрана. Лева половина текста није на екрану.	Центар текста је на горњој ивици екрана. Горња половина текста није на екрану.
position:0% align:end	Десна ивица текста је на левој ивици екрана. Цео текст није на екрану.	Доња ивица текста је на горњој ивици екрана. Цео текст није на екрану.
position:0%,line-left align:end	Лева ивица текста је на левој ивици екрана.	Горња ивица текста је на горњој ивици екрана.
position:50% align:start	Лева ивица текста је на хоризонталном центру екрана.	Горња ивица текста је на вертикалном центру екрана.
position:50% align:center	Центар текста је на хоризонталном центру екрана.	Центар текста је на вертикалном центру екрана.
position:50% align:end	Десна ивица текста је на хоризонталном центру екрана.	Доња ивица текста је на вертикалном центру екрана.
position:100% align:start	Лева ивица екрана је на десној ивици екрана. Цео текст је ван екрана.	Горња ивица текста је на доњој ивици екрана. Цео текст је ван екрана.
position:100%,line-right align:start	Десна ивица текста је на десној ивици екрана.	Доња ивица текста је на доњој ивици екрана.
position:100% align:center	Центар текста је на десној ивици екрана. Десни део текста је ван екрана.	Центар текста је на доњој ивици екрана. Доња половина текста је ван екрана.
position:100% align:end	Десна ивица текста је на десној ивици екрана.	Десна ивица текста је на десној ивици екрана.

3.3.3.4. Ред (енгл. *line*)

Подешавање се задаје командом *line*, док се вредност може дати као број или као проценат. Ефекат подешавања зависи од подешавања правца текста. Вредност подешавања може бити цео број (позитиван или негативан) или као проценат од 0 до 100, са знаком за проценат (%) Ефекат је дат у табели испод [9].

Табела 3 Ефекат подешавања линије у зависности од правца и поравњања [9]

Подешавање	Хоризонтални правац	Вертикални правац („lr“)	Вертикални правац („rl“)
line: 2, start	Горња ивица текста се налази две линије испод горње ивице екрана.	Лева ивица текста се налази две линије од леве ивице екрана.	Десна ивица текста се налази две линије од десне ивице екрана.
line: 2, center	Центар текста се налази две линије испод горње ивице екрана.	Центар текста се налази две линије од леве ивице екрана.	Центар ивица текста се налази две линије од десне ивице екрана.
line: 2, end	Доња ивица текста се налази две линије испод горње ивице екрана.	Десна ивица текста се налази две линије од леве ивице екрана.	Лева ивица текста се налази две линије од десне ивице екрана.
line: -2, start	Горња ивица текста се налази две линије изнад доње ивице екрана.	Десна ивица текста се налази две линије од десне ивице екрана.	Лева ивица текста се налази две линије од леве ивице екрана.
line: -2, center	Центар текста се налази две линије изнад доње ивице екрана.	Центар текста се налази две линије од десне ивице екрана.	Центар текста се налази две линије од леве ивице екрана.
line: -2, end	Доња ивица текста се налази две линије изнад доње ивице екрана.	Лева ивица текста се налази две линије од десне ивице екрана.	Десна ивица текста се налази две линије од леве ивице екрана.
line: percentage, start	Горња ивица текста се налази <i>percentage</i> растојања (од вертикалне дужине екрана) од горње ивице екрана.	Лева ивица текста се налази <i>percentage</i> растојања (од хоризонталне дужине екрана) од леве ивице екрана.	Десна ивица текста се налази <i>percentage</i> растојања (од хоризонталне дужине екрана) од десне ивице екрана.
line: percentage, center	Центар текста се налази <i>percentage</i> растојања (од вертикалне дужине екрана) од горње ивице екрана.	Центар текста се налази <i>percentage</i> растојања (од хоризонталне дужине екрана) од леве ивице екрана.	Центар ивица текста се налази <i>percentage</i> растојања (од хоризонталне дужине екрана) од десне ивице екрана.
line: percentage, end	Доња ивица текста се налази <i>percentage</i> растојања (од вертикалне дужине екрана) од горње ивице екрана.	Десна ивица текста се налази <i>percentage</i> растојања (од хоризонталне дужине екрана) од леве ивице екрана.	Лева ивица текста се налази <i>percentage</i> растојања (од хоризонталне дужине екрана) од десне ивице екрана.

3.3.3.5. Величина (енгл. *size*)

Подешавање се задаје именом *size*, а вредност може бити проценат у распону од 0 до 100, са укљученим знаком за проценат (%). Подешавање представља простор који заузима текст у односу на ширину екрана уколико је правац текста хоризонталан, односно у односу на висину екрана уколико је правац текста вертикалан [9].

3.3.3.6. Регион (*енгл. region*)

Сваки блок са текстом може бити придружен региону. Подешавање се задаје именом `region`. Вредност подешавања је идентификатор региона. Уколико се зада име региона које не постоји, подешавање ће бити игнорисано [9].

3.3.4. Текст

У овом делу блока налази се садржај који се приказује на екрану. Овај садржај се налази ред испод временске ознаке и може се простирати у више редова. Прелази у нове редове ће бити очувани приликом приказивања садржаја на видеу. Уколико садржај не може да стане у једну линију приликом приказивања, биће распоређен у више линија. У овом делу блока уз садржај се могу наћи разнолике ознаке које служе за стилизовање текста.

3.3.4.1. Стилизовање помоћу ознака

Текст је могуће форматирати помоћу ознака. Свакој ознаци се могу придружити класе које треба такође да се примене на текст који се налази у датој ознаци. Класе су међусобно раздвојене тачком. Постоје подразумеване класе за боју текста и боју позадине. Уколико је потребно специфично стилизовање, могуће је дефинисати произвољну класу помоћу *CSS-a*. Ознаке које су подржане ће бити описане у наставку. Све ознаке могу да се отворе и затворе. Кад се ознака отвори њен ефекат, класе и остали стилови се примењују на текст који се налази у њој до затварања ознаке. [2] Ознаке се задају по формату на следећој слици:

```
<име_ознаке.додатне_класе атрибут_ако_ознака_има_атрибут> текст или друге ознаке </име_ознаке>
```

Слика 9 Формат коришћења ознака

- **Гласовна ознака (*енгл. voice tag*)**

Гласовна ознака се задаје именом `v` и има атрибут који означава ко изговара текст. Пример употребе се налази на следећој слици:

```
<v Душан> Данас је леп дан. </v>
```

Слика 10 Пример употребе гласовне ознаке

- **Ознака за курсив (*енгл. italic tag*)**

Курсив ознака се задаје именом `i` и нема додатне атрибуте. Пример употребе се налази на следећој слици:

```
<i> Овај текст ће бити написан у курсиву </i>
```

Слика 11 Пример употребе ознаке за курсив

- **Ознака за подебљање (енгл. *bold tag*)**

Ознака за подебљање се задаје именом **b** и нема додатне атрибуте. Пример употребе се налази на следећој слици:



```
<b> Овај текст ће бити подебљан </b>
```

Слика 12 Пример употребе тага за подебљање текста

- **Ознака за подвлачење (енгл. *underline tag*)**

Ознака за подвлачење текста се задаје именом **u** и нема додатне атрибуте. Пример употребе се налази на следећој слици:

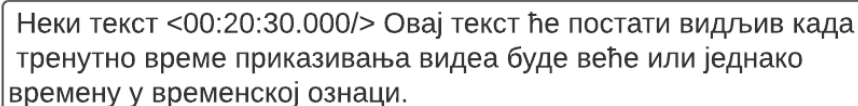


```
<u> Овај текст ће бити подвучен </u>
```

Слика 13 Пример употребе ознаке за подвлачење текста

- **Временска ознака (енгл. *timestamp tag*)**

Задаје се у формату као и временска ознака за блок превода. Време у ознаци мора бити веће или једнако од почетног времена приказивања текста и мање или једнако од завршног времена приказивања текста. Текст који се налази са десне стране ознаке ће се приказати тек када време видеа буде веће или једнако времену у ознаку. Пример употребе се налази на следећој слици:



```
Неки текст <00:20:30.000/> Овај текст ће постати видљив када  
тренутно време приказивања видеа буде веће или једнако  
времену у временској ознаци.
```

Слика 14 Пример употребе временске ознаке

- **Ознака за језик (енгл. *language tag*)**

Ознака за језик се задаје именом **lang** и има атрибут који говори о ком језику се ради. Пример употребе се налази на следећој слици:



```
<lang en-US> Where? </lang>
```

Слика 15 Пример употребе ознаке за језик

- **Ознака за класу (енгл. *class tag*)**

Ознака за класу се задаје именом **c** и нема додатне атрибуте. Нема посебан ефекат као остали тагови. Смисао употребе је да се додају класе на неки део текста без додавања још неког понашања. Пример употребе се налази на следећој слици:

```
<c> Неки текст </c>  
<c.ime_klase> Неки други текст </c>
```

Слика 16 Пример употребе ознаке за класу

- **Текст изнад текста (енгл. *ruby tag*)**

Ознака се користи уколико је потребно ставити нешто изнад текста. Прво се почиње ознаком `ruby`, а затим се задаје база (сам текст), након које помоћу ознаке `rt` се задаје текст који ће бити стављен изнад базе. Овај образац се понавља колико год пута желимо. Пример употребе и ефекат се налази на следеће две слике, респективно.

```
<ruby> neki tekst <rt> nesto gore1 </rt> nesto drugo <rt> nesto gore  
2 </rt> </ruby>
```

Слика 17 Пример употребе тага за горњи део текста

текст изнад
база

Слика 18 Ефекат ознаке за текст изнад текста

3.3.4.2. *HTML* референце за карактере

У тексту превода се могу наћи и *HTML* ознаке које означавају неке карактере или скуп карактера. Почињу знаком `&` и завршавају се знаком `;`. Могу бити:

- Именоване (пример: *equals*)

Задају се помоћу низа слова. По стандарду постоји табела [14] која мапира низ карактера на вредност карактера у уникоду. Задати низ карактера се замењује уникод вредношћу карактера. Уколико се низ карактера није пронашао у табели, задати низ се оставља у тексту.

- Хексадецималне

Након знака `&` налази се знак `#`, након којег се налази мало или велико латинично слово `x`. Затим је потребно навести једну или више хексадецималних цифара. Број се интерпретира у хексадецималном систему и добијена вредност представља уникод вредност која се замењује у текст уместо наведене референце. Пример употребе се налази на следећој слици:

```
Ово ће бити замењено са знаком једнако: &#x0003D;
```

Слика 19 Пример употребе хексадецималне референце

- Децималне

Након знака `&` налази се знак `#`. Затим следи једна или више децималних цифара. Број се интерпретира у децималном систему и добијена вредност представља уникод вредност која се замењује у текст уместо наведене референце.

Ово ће бити замењено са знаком једнако: `=`;

Слика 20 Пример употребе децималне референце

3.4. Регион

Региони (*енгл. region*) омогућавају да се направи специфична област којом се дефинише локација, геометрија, као и стилови који ће се примењивати на текстове који су придружени региону. Када се регион направи, могуће му је доделити блокове са текстом помоћу подешавања `region`, приликом дефинисања блока са текстом. Подешавања региона се задају у формату `име:вредност`. Она морају бити међусобно одвојена белим знаком. Региони се могу дефинисати само за хоризонтални правац текста. Пример подешавања региона може се видети на следећој слици: [9]

```
REGION
id:пример
width:50%
lines:4
regionanchor:0%,100%
viewportanchor:10%,90%
scroll:up
```

Слика 21 Пример подешавања региона

У наставку ће бити објашњена могућа подешавања региона.

3.4.1. Идентификатор (*енгл. id*)

Идентификатор представља јединствено име региона. Задаје се командом `id`. Вредност представља јединствено име региона. Идентификатор се користи приликом додавања блока са текстом у регион, као и приликом стилизовања региона.

3.4.2. Ширина (*енгл. width*)

Ширина представља ширину блока која представља регион, Задаје се командом `width`. Вредност се изражава у процентима од 0 до 100, после које се налази знак за проценат (%). Вредност представља ширину региона у односу на ширину екрана. [9]

3.4.3. Број линија (*енгл. lines*)

Број линија представља висину региона у линијама. Задаје се именом `lines`. Вредност је цео број и мора бити позитивна. [9]

3.4.4. Начин додавања новог блока са преводом

Подешавање се задаје именом `scroll`. Вредност може бити само `up`. Вредност `up` значи да ће се стари блокови померати ка горњој ивици региона и полако нестајати како додајемо нове

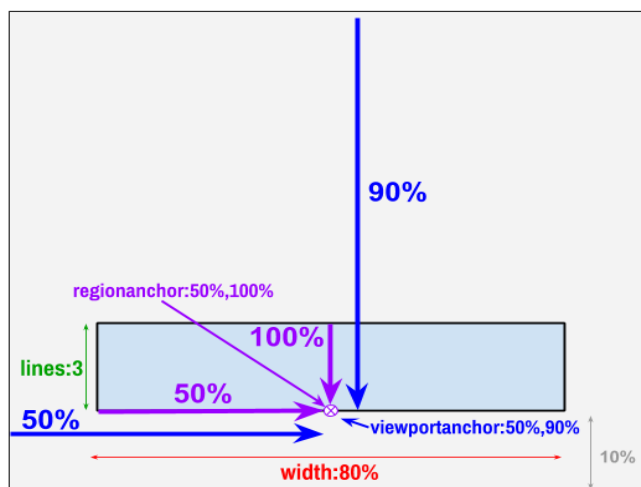
блокове са текстом. Уколико подешавање није наведено, онда се стари блокови са текстом неће померати са своје оригиналне позиције, већ ће се нови блокови са текстом додавати испод последњег блока, и у неком тренутку ће се нови блокови додавати у део региона који није видљив на екрану. Пошто је величина региона фиксирана, текст који не може да стане се одсеца. Овим подешавањем се говори да ли се одсеца блок са текстом на горњој ивици или на доњој ивици региона, односно да ли се одсеца блок са текстом који се већ налази у региону или блок који се тек додаје у регион. [2]

3.4.5. Тачка која представља регион (енгл. *region anchor*)

Ово подешавање дефинише координате које ће представљати регион. Задаје се командом `regionanchor`, док се вредност задаје као два знака за проценат раздвојена запетом. Прва вредност дефинише *x* координату у односу на ширину целог региона, док друга дефинише *y* координату у односу на висину целог региона. [9]

3.4.6. Позиција региона на екрану (енгл. *view port anchor*)

Подешавање дефинише координате тачке која преставља регион на екрану, а самим тим и позицију региона. Задаје се командом `viewportanchor`, док се вредност задаје као два знака за проценат раздвојена запетом. Прва вредност дефинише *x* координату у односу на ширину екрана, док друга дефинише *y* координату у односу на висину екрана. Објашњење подешавања може се видети на слици: [9]



Слика 22 Објашњење подешавања `regionanchor` и `viewportanchor` [2]

3.5. Стили (енгл. *stylesheets*)

За стилизовање блокова са текстом и региона могуће је користити *CSS*. *WebVTT* формат је увео посебне псеудо-елементе (`::cue` и `::cue-region`). У наставку ће бити објашњено на који начин се могу користити ови псеудо-елементи. Могуће је и писање коментара као у *CSS*-у (`/* коментар*`). [2]

3.5.1. Стили за регионе (енгл. `::cue-region`)

Код стилова региона, приликом задавања селектора, постоје две опције. Једна опција је идентификатор региона (пре имена треба ставити знак `#`) и то означава да се стилови

примењују само на регион за задатим именом, док је друга опција без заграда и то значи да се задати стилови примењују на све регионе. [2]

```
STYLE
::cue-region(#име-региона1),
::cue-region(#име-региона2) {
    color: blue;
    ...
}
/*Биће примењено на регион са
идентификатором "име-региона3"*/
::cue-region(#име-региона3) {
    color: black;
}
/*Ово ће бити примењено на све регионе */
::cue-region{
    background-color: yellow
}
```

Слика 23 Пример стилизовања региона

3.5.2. Стили за блокове са текстом (енгл. *::cue*)

Код стилова за блокове са текстом, могуће је задати било који селектор који је дефинисан CSS документацијом. Приликом примене стилова, стабло које се формира помоћу текста превода се понаша као *DOM* [15] у *HTML*-у, док се ознаке понашају као *HTML* ознаке. У наставку ће бити дати примери на који начин се могу користити селектори и подешавати стилови.

3.5.2.1. Стил за све блокове са текстом

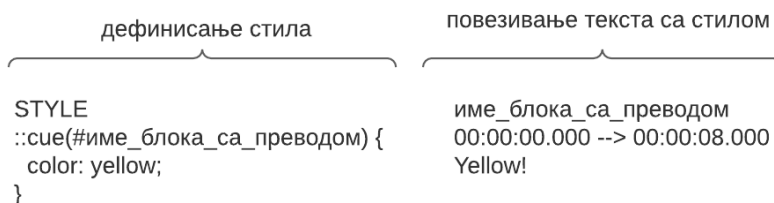
Стил који се зада на овакав начин ће бити примењен на све блокове са текстом. Пример коришћења се налази на слици испод. [2]

```
STYLE
::cue {
    color: yellow;
}
```

Слика 24 Стил који се примењује на све блокове са текстом

3.5.2.2. Идентификатор селектор

Стил који је задат овим селектором се примењује на блок са преводом који има исто име као и наведено име у селектору. На слици испод се може видети начин употребе, са леве стране се налази дефинисани стил, а са десне блок са преводом за који ће се применити дефинисани стилу. [2]



Слика 25 Пример стилизовања блока са текстом помоћу идентификатора

3.5.2.3. Селектор по типу ознаке

Стил који је задат неким селектором овог типа се примењује на делове текста у блоку који се налазе у ознаци са истим именом као што је име селектора. На слици испод се налази пример употребе као и дозвољена имена за име селектора (иста имена као и за имена ознака). [2]

дефинисање стила	повезивање текста са стилем
<pre>:cue(c), :cue(i), :cue(b), :cue(u), :cue(ruby), :cue(rt), :cue(v), :cue(lang) { color: yellow; }</pre>	<pre>00:00:00.000 --> 00:00:08.000 <c>Yellow!</c> <i>Yellow!</i> <u>Yellow!</u> Yellow! <u>Yellow!</u> <ruby>Yellow! <rt>Yellow!</rt></ruby> <v Kathryn>Yellow!</v> <lang en>Yellow!</lang></pre>

Слика 26 Пример стилизовања блока са текстом помоћу селектора по типу ознаке

3.5.2.4. Селектор по имену класе

Стил који се дефинише преко селектора класе ће се применити на делове текста који се налазе у некој ознаци којој је додељена та класа (не морају бити директно у тој ознаци, могу се налазити и у оквиру ознаке која се налази у оквиру ознаке са датом класом на произвољној дубини). Класа се може доделити било којој ознаци. Пример употребе се налази на следећој слици. [2]

<pre>::cue(.loud) { color: yellow; }</pre>	<pre>00:00:00.000 --> 00:00:08.000 <c.loud>Yellow!</c> <i.loud>Yellow!</i> <u.loud>Yellow!</u> <b.loud>Yellow! <u.loud>Yellow!</u> <ruby.loud>Yellow! <rt.loud>Yellow!</rt></ruby> <v.loud Kathryn>Yellow!</v> <lang.loud en>Yellow!</lang></pre>
--	--

Слика 27 Пример стилизовања блока са текстом помоћу селектора класе

3.5.2.5. Селектор по атрибуту

Стил који се дефинише овим селектором ће се применити на све ознаке са истим именом и истом вредношћу атрибута као у наведеном селектору. У време писања овог документа подржана су два типа селектора са атрибутом (као и две ознаке): `voice` и `lang`. Ознаке који имају ове атрибуте су `v` и `lang` респективно. За текст се може подесити подразумевани језик. Ефекат постављања подразумеваног ефекта је еквивалентан стављањем текста у `lang` ознаку, са атрибутом чија је вредност једнака постављеном језику. [2]

video::cue([lang="en-US"]) {	00:00:00.000 --> 00:00:08.000
color: yellow;	Yellow!
}	
video::cue([lang="en-GB"]) {	00:00:08.000 --> 00:00:16.000
color: cyan;	<lang en-GB>Cyan!</lang>
}	
video::cue(v[voice="Kathryn"]) {	00:00:16.000 --> 00:00:24.000
color: lime;	<v Kathryn>I like lime.</v>
}	

Слика 28 Пример стилизовања блока са текстом помоћу селектора са атрибутом

3.5.2.6. Псеудо-класа селектор

Селектори у виду псеудо-класе почињу знаком **:**, иза кога следи име селектора. Сваки селектор има своје значење. Како постоји велики број псеудо-класа [16], овде ће бити описане неке специфичне (оне које су специфичне за формат *WebVTT-a*), док се остале могу наћи у документацији *CSS-a*. [2]

- **Future псеудо-селектор**

Стил који је дефинисан овим селектором се примењује на део текста који се налази са десне стране временске ознаке, уколико је време у ознаци веће од тренутног времена видеа који се приказује.

- **Past псеудо-селектор**

Стил који је дефинисан овим селектором се примењује на део текста који се налази са леве стране временске ознаке, уколико је време у ознаци мање од тренутног времена видеа који се приказује.

На следећој слици се може видети пример употребе. Ови псеудо-селектори се најчешће користе за караоке. Помоћу њих се може означити текст који је отпеван (*past*), текст који тек треба да се отпева (*future*) као и текст који се тренутно пева. [9]

	WEBVTT - Пример употребе :future и :past селектора
	1
	00:16.500 --> 00:18.500
	When the moon <00:17.500>hits your eye
STYLE	
::cue:past { color:yellow }	2
::cue:future { text-shadow: black 0 0 1px; }	00:00:18.500 --> 00:00:20.500
	Like a <00:19.000>big-a <00:19.500>pizza <00:20.000>pie
	3 00:00:20.500 --> 00:00:21.500
	That's <00:00:21.000>amore

Слика 29 Пример употребе future и past псеудо селектора

3.5.2.7. Псеудо-елемент селектор

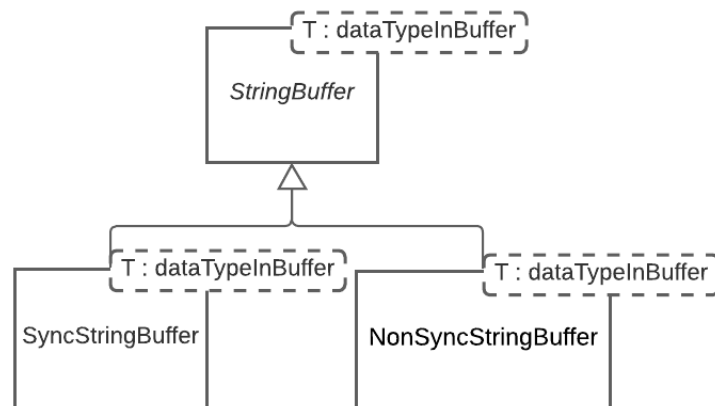
Псеудо-елемент селектор се задаје помоћи два знака **:** после којих се наводи име псеудо-елемента. У стандарду *CSS-a* постоји велики број различитих псеудо-елемената [17]. Стандард *WebVTT-a* прописује два псеудо-елемента (*::cue* и *::cue-region*). Међутим, псеудо-елементи се могу користити и као селектор. Стандард *WebVTT* не описује ниједан специфичан селектор који је псеудо-елемент.

4. Имплементација

У овом поглављу ће бити објашњено на који начин је имплементиран парсер *WebVTT* формата и биће изнети неки специфични детаљи имплементације.

4.1. Бафер за стринг

Пошто проблем парсирања формата за видео титлове спада у проблеме у реалном времену, врло вероватно неће бити доступан цео фајл приликом парсирања, па зато морамо да се бавимо синхронизацијом и чекамо док не дође још садржаја фајла. Могуће је имати цео фајл одмах приликом пуштања видеа и онда нема потребе да се бавимо синхронизацијом. Из овог разлога направљена је основна класа *StringBuffer*, која представља основну класу која садржи декларације метода које су неопходне за рад парсера. Пошто у нашем систему радимо са *UTF-8* и *UTF-32* стринговима, направљен је шаблон класе (енгл. *template*) који прима тип стринга који се чува у баферу. Из ове класе се изводе *SyncStringBuffer*, који имплементира синхронизацију и чекање података у својим методама и *NonSyncStringBuffer*, који нема синхронизацију, већ само довлачи податке из бафера. Приликом коришћења бафера који нема подршку за синхронизацију, корисник који користи систем је дужан да се побрине да пре покретања парсирања стави све податке у њега и да тек онда покрене парсер. Парсер прима *StringBuffer*, тако да корисник може да проследи коју год имплементацију жели. На следећој слици се налази упрошћени класни дијаграм.

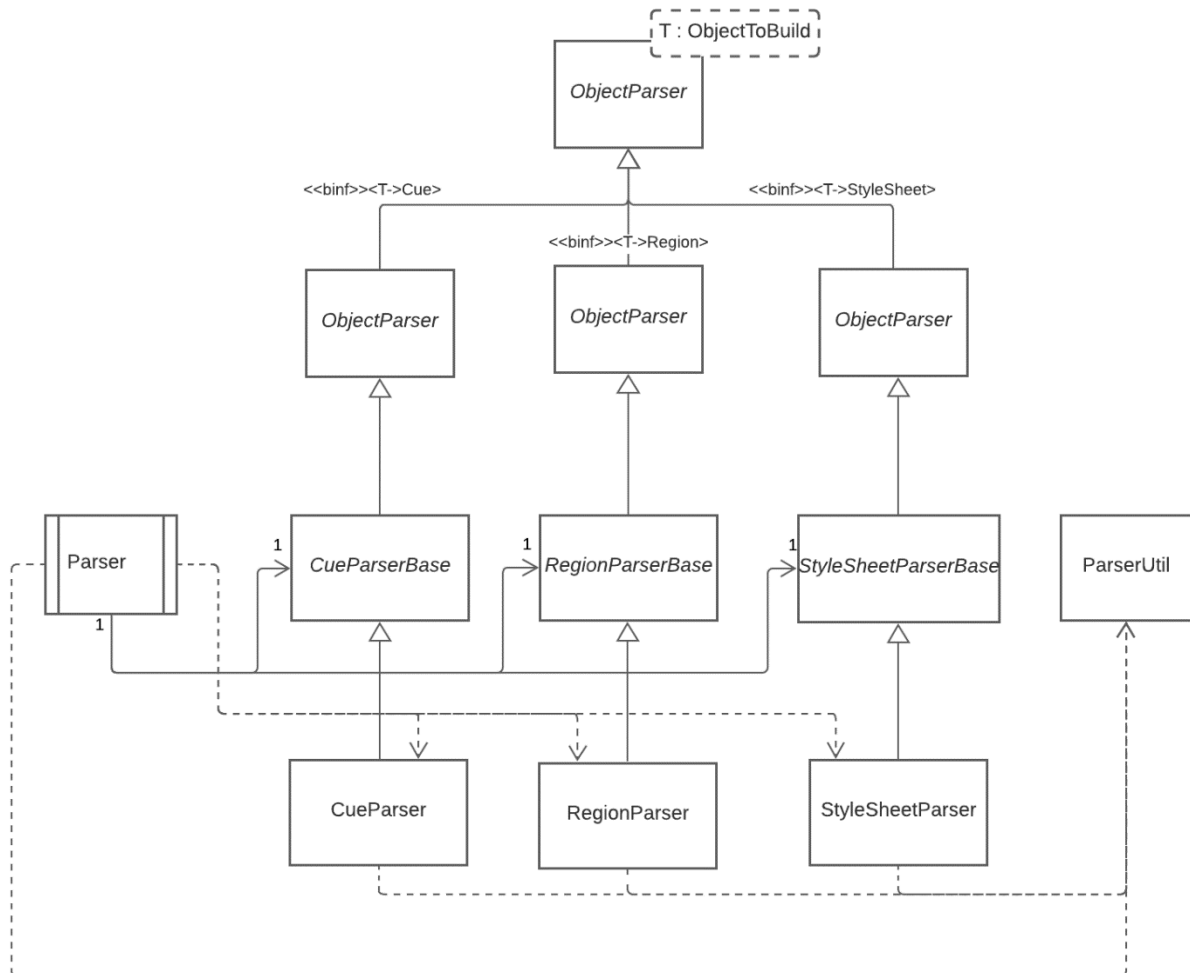


Дијаграм класа 1 Бафер

4.2. Главни део парсера

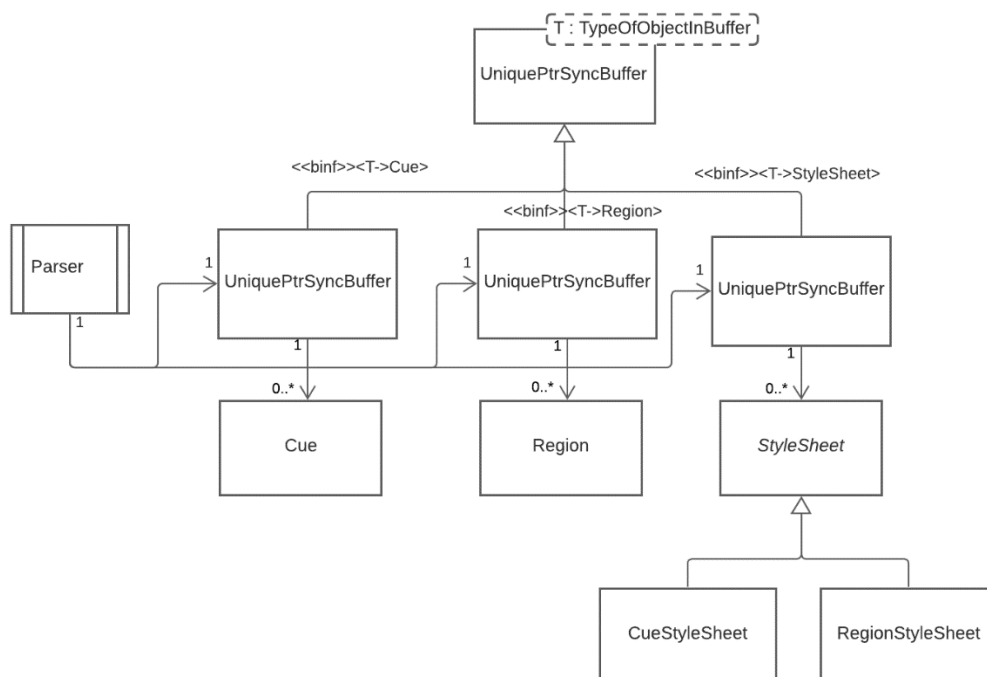
Парсер прима *StringBuffer* из кога узима податке и парсира их. Корисник се може одредити да користи *StringSyncBuffer* имплементацију или *NonSyncStringBuffer* имплементацију. Парсер као излаз даје бафер блокова са текстом, бафер региона и бафер стилова. Сви ови бафери су направљени помоћу шаблона класе *UniquePtrSyncBuffer*. Слично као и бафер са стринговима, *UniquePtrSyncBuffer* има у себи имплементирану синхронизацију, тако да корисник може одмах да користи бафере чим започне парсирање. Сам парсер поседује специјализоване парсере, који се баве парсирањем сваког блока посебно. Специјализовани парсери добију део стринга и из њега праве део одговарајућег блока. Објекти блокова који се праве из стринга имају сва поља која су дефинисана предлогом стандарда [18]. Пошто сви специјализовани парсери имају објекат који праве, као и могућност да се направи нов објекат, као и да се

преузме направљени објекат, направљен је шаблон класе са тим могућностима. Шаблон класа прима параметар који је типа блока који се парсира. Сваки тип парсера представља изведену класу из шаблонске класе са одговарајућим параметром. Парсер има главну петљу у којој одлучује о ком типу објекта се ради и на основу тога позива одговарајући парсер. Сви парсери користе услужну класу ParserUtil која имплементира разне методе за парсирање и обраду стринга. На слици испод се налази упрошћен дијаграм класа, који представља структуру парсера.



Дијаграм класа 2 Структура парсера

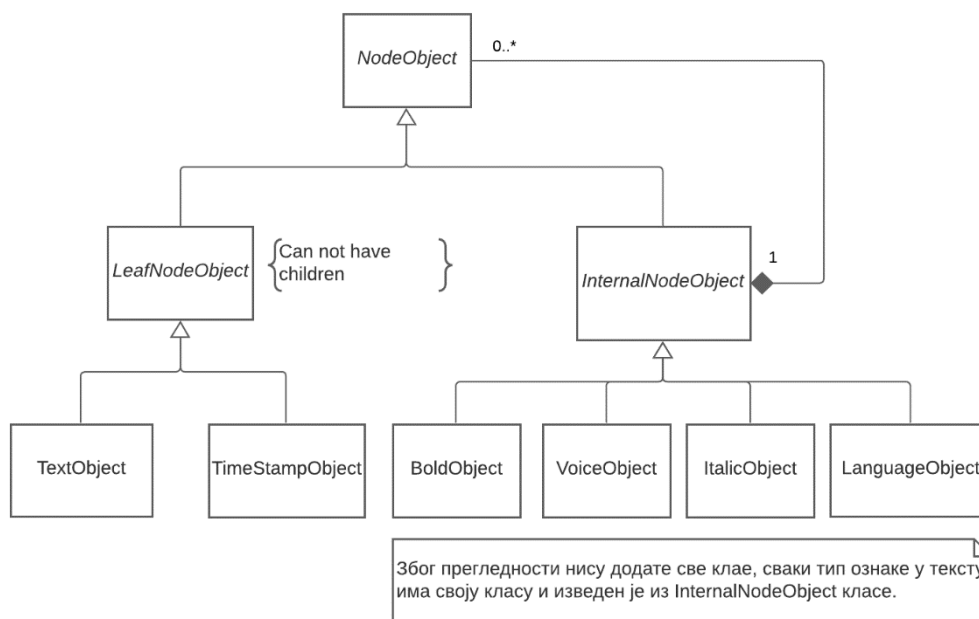
Тренутно парсер зависи од конкретних имплементација специфичних парсера, јер их прави у свом конструктору. Следећи корак би био да се направи неки други начин за постављање зависности парсера (прављење специфичних парсера). На пример, то је могуће урадити увођењем методе за постављање одговарајућег типа парсера. У том случају, метода би примала CueParserBase и у њој би се поставио специфични парсер на одговарајућу вредност. У тренутној имплементацији система ово није урађено, јер је сматрано да би се закомпликовало само коришћење парсера, а тренутно нема потребе за различитим врстама специфичних парсера. Излаз парсера представљају бафери блокова. Кориснику се препоручује да погледа изворни код како би се упознао са могућностима које нуде поменути бафери. На следећој слици се налази упрошћен дијаграм класа, који представља структуру излаза парсера.



Дијаграм класа 3 Структура излаза парсера

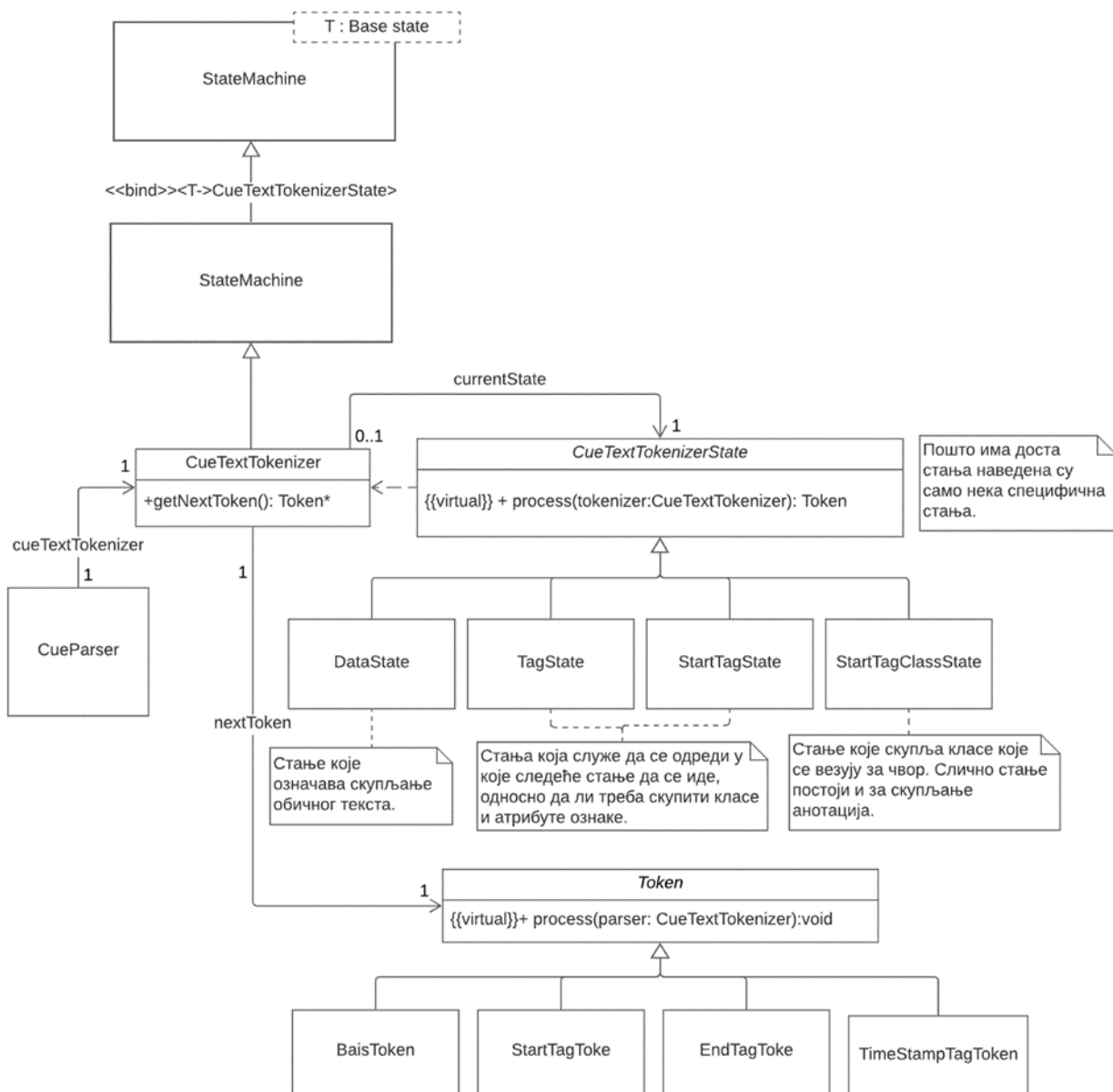
4.3. Парсирање блока са текстом

Како се у тексту могу наћи многе ознаке које означавају примену одговарајућег стила, за текст превода се прави стабло, тако да сви делови текста који припадају једној ознаци буду деца тог чвора стабла. Структура стабла је направљена помоћу пројектног узорка Састав (*енгл. Composite*) [19]. Унутрашњи чворови стабла могу имати децу и сваки тип ознаке који постоји има свој чвор. Листови стабла не могу имати децу и они могу бити или сам текст, или временска ознака. Структура стабла се налази на слици испод.



Дијаграм класа 4 Структура стабла за текст превода

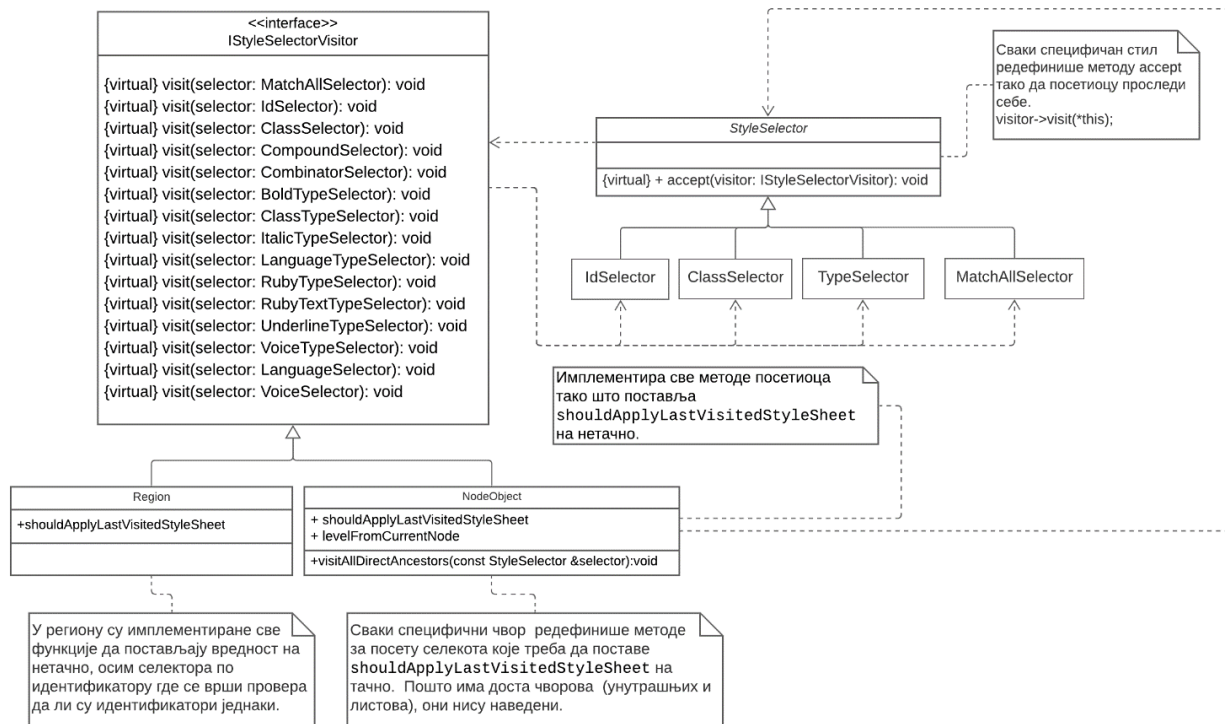
Како би се на најлакши начин направило стабло, извршена је трансформација улазног текста на токене помоћу аутомата стања (*енгл. State machine*) [20]. Сваки токен зна шта треба да уради, односно на који начин треба да утиче на изглед стабла. За трансформацију улазног текста и прављења токена имплементиран је алгоритам који се налази у предлогу стандарда. [1] Идеја парсирања је да `CueParser` позива методу за дохватање новог токена од `CueTextTokenizer`-а и да затим над тим токеном позове методу за његову обраду. Парсер не зна ког је конкретно типа токен. Метода за обраду токена је редефинисана у сваком типу токена. `CueTextTokenizer` је аутомат стања и прави следећи токен помоћу стања која поседује. Он позива методу за обраду над текућим стањем, све док му позвана метода не врати неки токен (вредност која није `nullptr`). Након што је добио токен, прослеђује га позиваоцу. `CueTextTokenizer` не зна конкретан тип текућег стања, он само над текућим стањем позива методу за обраду стања. У сваком конкретном стању редефинисана је метода за обраду стања. Стања аутомата зависе од конкретних врста токена, јер се токени праве у њима. Праве се помоћу пројектног узорка "Мува" (*енгл. Flyweight*) [21] и "Статичка фабричка метода" (*енгл. Static factory method*) [22]. Пројектни узорци су имплементирани у основној класи за сва стања. У основној класи за сва стања `CueTextTokenizerState` налази се статичка метода која прима тип објекта који се тражи. У методи се испитује да ли је некада већ направљен објекат датог типа и ако јесте враћа се тај тип, а уколико није прави се нови и он се враћа. Како би се обезбедило да стања могу безбедно да се деле између различитих аутомата, направљена су тако да немају унутрашње стање. Метода за обраду стања прима аутомат стања, за који треба да обави посао. На следећој слици се налази структура дела парсера који се бави прављењем стабла.



Дијаграм класа 5 Парсирање превода видеа

4.4. Парсирање блока са стилем

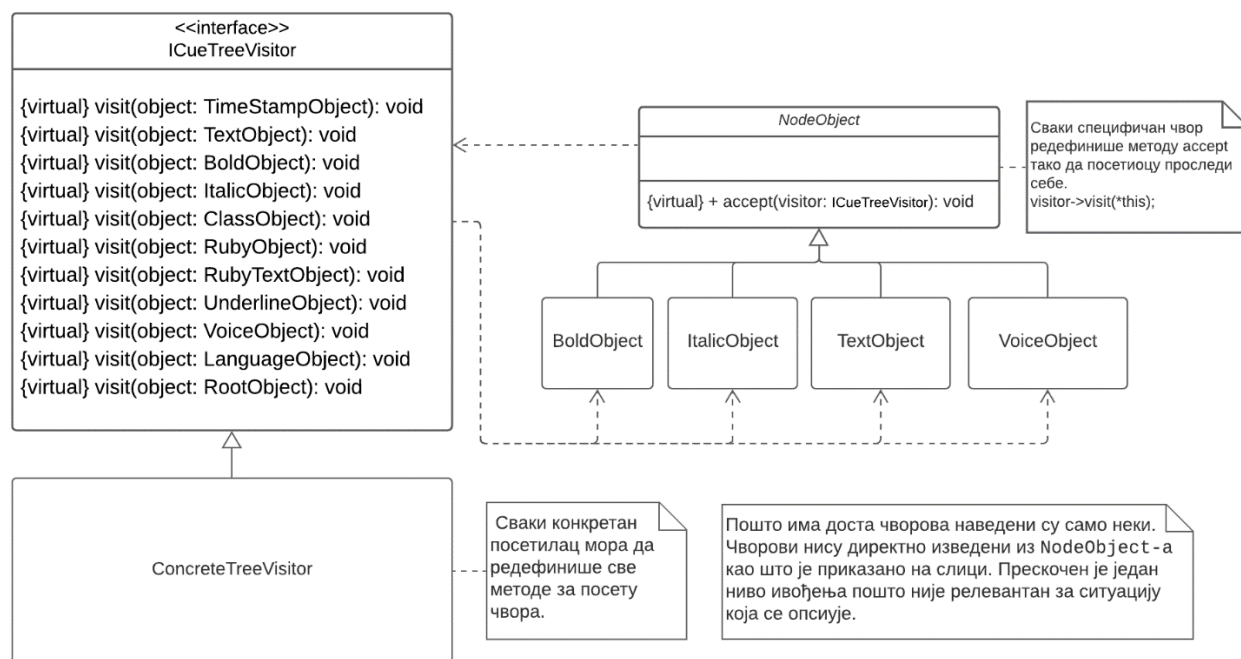
За парсирање блока са стилем направљен је посебан аутомат стања **Stylesheet Parser** који прави одговарајући **Stylesheet** и додељује му одговарајуће селекторе. За сваки тип селектора направљена је посебна класа, како би се лакше одлучило да ли се неки селектор поклапа са регионом или чвором стабла. **Stylesheet** поседује мапу која садржи имена атрибута (и вредности) које треба применити уколико се селектор поклапа са прослеђеним регионом или чвором стабла. Слично као код парсирања блока са текстом, стања аутомата су направљена помоћу пројектног узорка "Мува" и статичког фабричког метода. Пројектни узорци су имплементирани у основној класи за сва стања. На следећој слици налази се упрошћени дијаграм класа.



Дијаграм класа 7 Структура потребна за проверу поклапања селектора

4.6. Подршка за обилазак стабла превода

Како бисмо исписали текст превода на екран, потребно је да обиђемо стабло превода. Приликом обиласка стабла ће се користити и механизам објашњен у претходном поглављу, како би се одредило који стилови треба да се прикажу на који део текста. Уколико клијент подржава само испис текста, а не и његово стилизовање, онда није потребно користити претходни механизам, него само обићи стабло и покупити текст из чворова листова који су типа TextObject. Како би се подржао обилазак стабла, користи се пројектни узорак "Посетилац". Направљен је интерфејс који садржи по једну декларацију методе за сваки тип чвора стабла који постоји. Сваки чвор стабла има методу за прихватање посетиоца и у тој методи над посетиоцем зове методу за посету чвора тако што му прослеђује себе. На тај начин ће у посетиоцу бити позвана метода која обилази баш тај конкретан чвор. Свака класа која жели да има могућност обиласка стабла треба да имплементира овај интерфејс. На следећој слици се налази класни дијаграм који имплементира објашњени механизам.



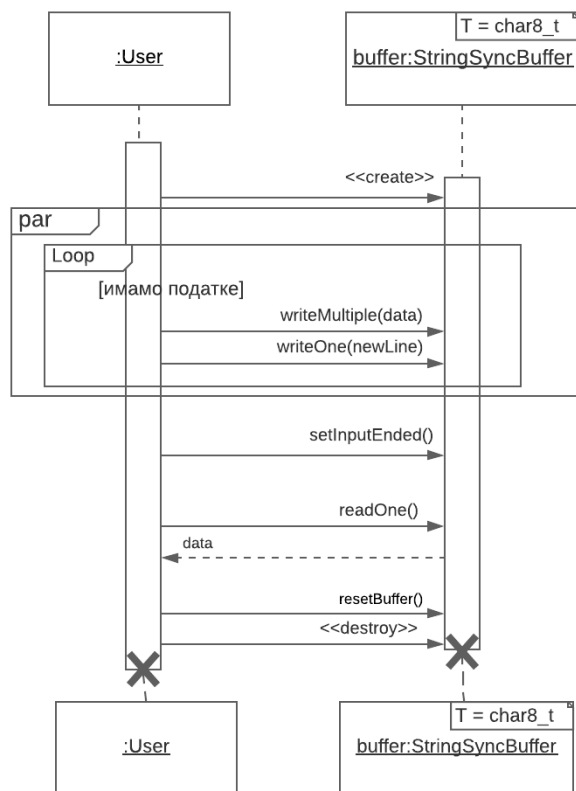
Дијаграм класа 8 Структура потребна за обилазак стабла превода

5. Корисничко упутство

У овом поглављу ће бити описан начин на који корисници могу користити систем.

5.1. Бафер за стринг

Овде ће бити објашњено на који начин корисник може користити бафер. Кориснику су на располагању два шаблона класе `StringSyncBuffer` и `NonSyncStringBuffer`, које имају заједничку основну класу `StringBuffer`. Прва класа имплементира синхронизацију, док је друга не имплементира. На следећој слици се налази пример употребе бафера. Корисник прво треба да направи одговарајући тип бафера, тако што ће инстанцирати одговарајући шаблон класе (у зависности од тога да ли му је потребна синхронизација) са типом податка који жели да складишти. Након прављења типа бафера и објекта тог типа, корисник може да користи бафер. Кориснику се препоручује да погледа `hrr` фајлове наведених класа како би видео све могућности које су му на располагању. На следећој слици се налази пример употребе бафера.



Дијаграм секвенце 1 Пример употребе бафера

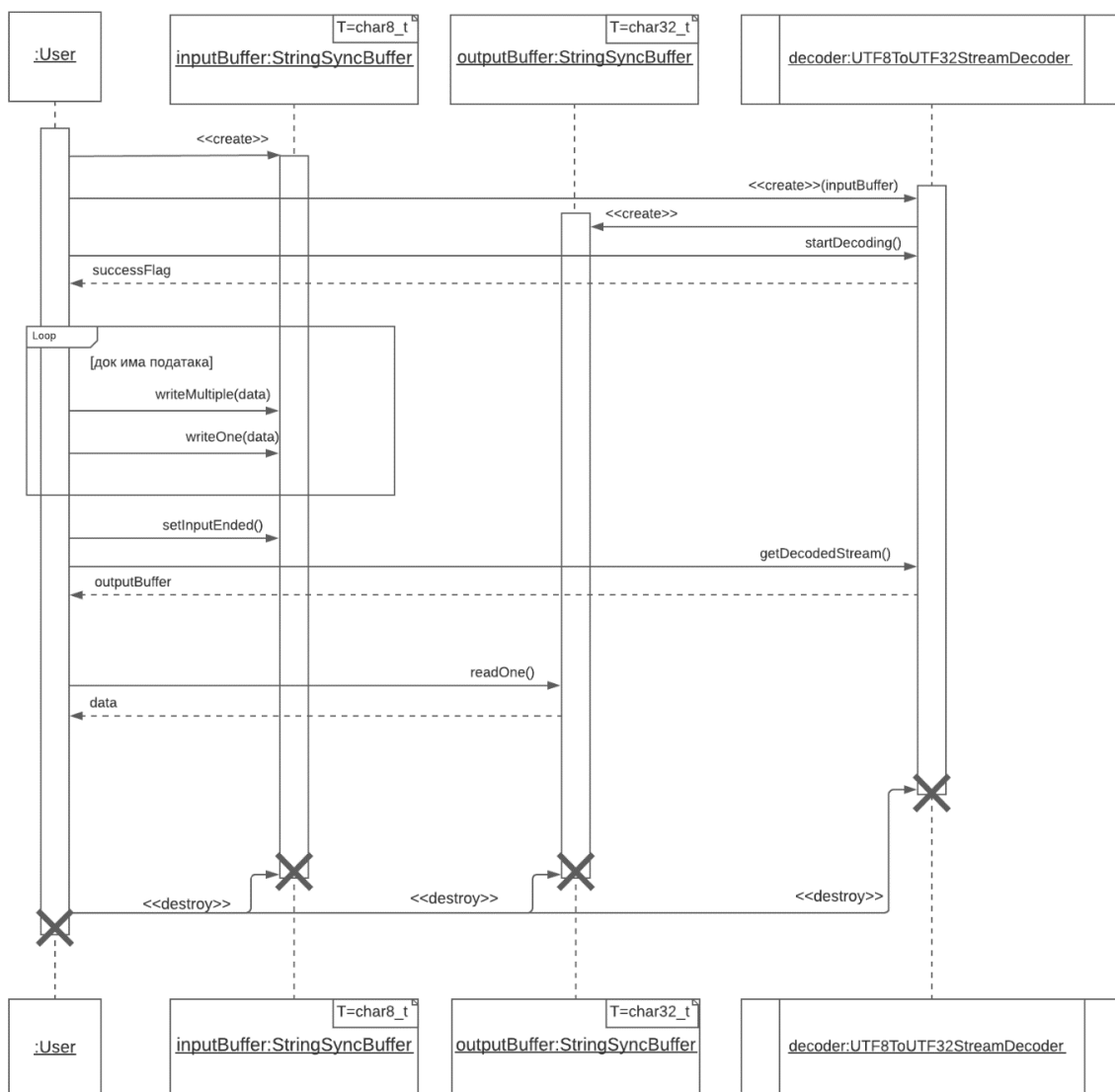
5.2. Бафер за блокове

Пошто парсер на излазу даје бафер за сваку врсту блока, направљен је шаблон класе са именом `UniquePtrSyncBuffer`. Шаблон класе прима параметар који означава тип елемената који се чувају у баферу. За разлику од бафера за стрингове, постоји само једна имплементација бафера, и то она која имплементира синхронизацију. Подаци се чувају помоћу паметних показивача, и то помоћу јединственог показивача. Корисник који користи бафер може да користи податке из њега и није одговоран за њихово брисање. Када се бафер буде брисао, он

ће обрисати и податке које се налази у њему. За списак метода које се могу позвати над бафером може се погледати изворни код.

5.3. Декодер

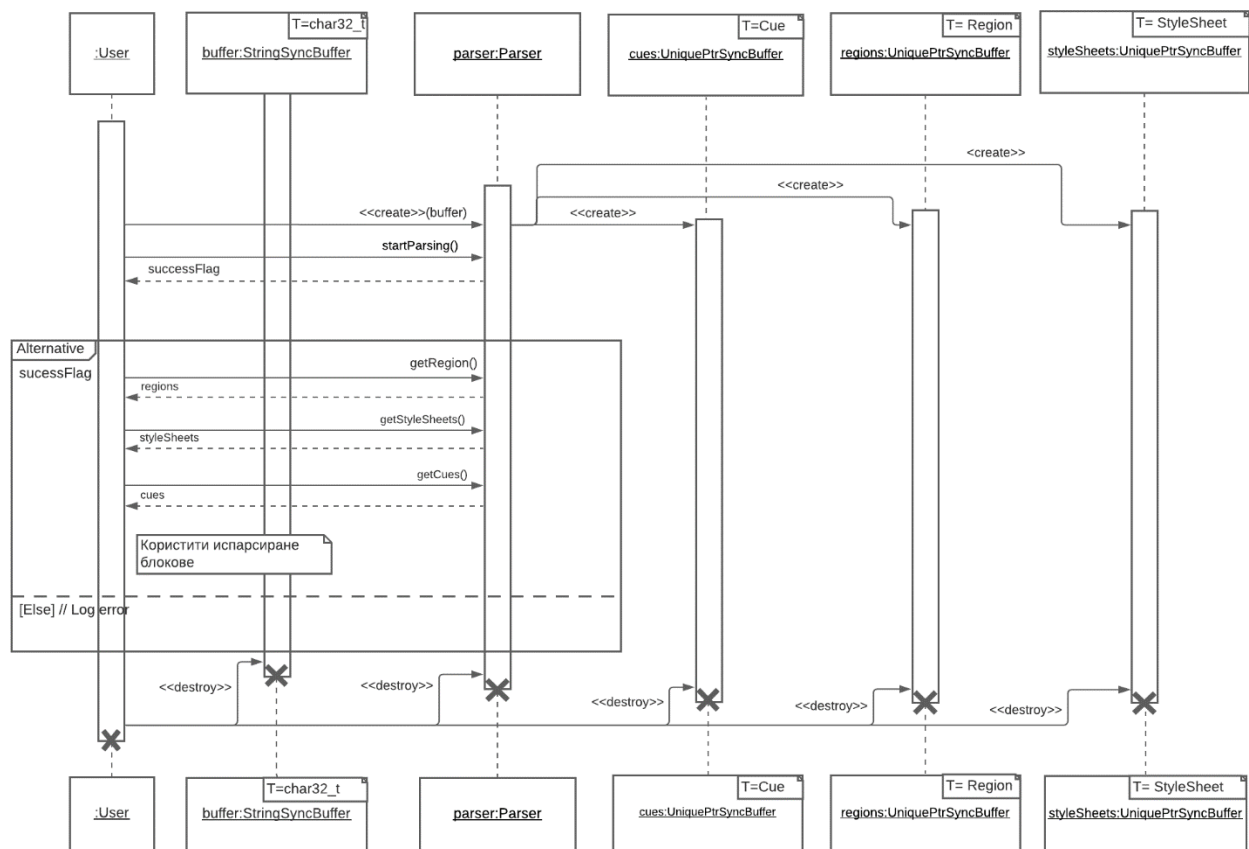
Како систем ради са *UTF-8* стринговима и како код њих један карактер може да буде дужине од 1 до 4 бајта, то би правило проблеме код обраде стрингова. Због тога је извршено пребацивање у *UTF-32* стринг, код кога је један карактер увек дужине 4 бајта. Декодер конвертује стринг из *UTF-8* енковања у *UTF-32* [23] енковање. Уколико је улазни текст *UTF-8* енкован, корисник система мора да га прво конвертује у *UTF-32*, пре него што проследи тај стринг парсеру. Декодер прима у конструктору бафер стринга који конвертује. Приликом стартовања декодовања, покреће се нит која врши декодовање и враћа се кориснику индикатор да ли је покретање успешно или не. На свом излазу декодер даје бафер типа *StringSyncBuffer*, који корисник може користи одмах након покретања декодера. Декодер ће, како буде вршио декодовање, убацивати у бафер декодоване податке. На слици испод се налази пример употребе декодера.



Дијаграм секвенце 2 Пример употребе декодера

5.4. Парсер

Парсер прима бафер из кога ће узимати податке и парсирати их. Слично као код бафера, корисник може да се одлучи за имплементацију бафера која њему одговара (синхронизовану или несинхронизовану). Корисник затим треба да покрене парсирање, и затим може да узме од парсера излазне бафере и да их користи. На следећој слици се налази пример употребе парсера. Претпоставка је да се у улазном баферу налази *UTF-32* стринг. Уколико је корисник имао *UTF-8* стринг, потребно је да позове декодер како би добио *UTF-32* стринг и да затим излазни бафер декодера проследи парсеру.



Дијаграм секвенце 3 Пример употребе парсера

За скуп могућих операција са баферима блокова погледати изворни код.

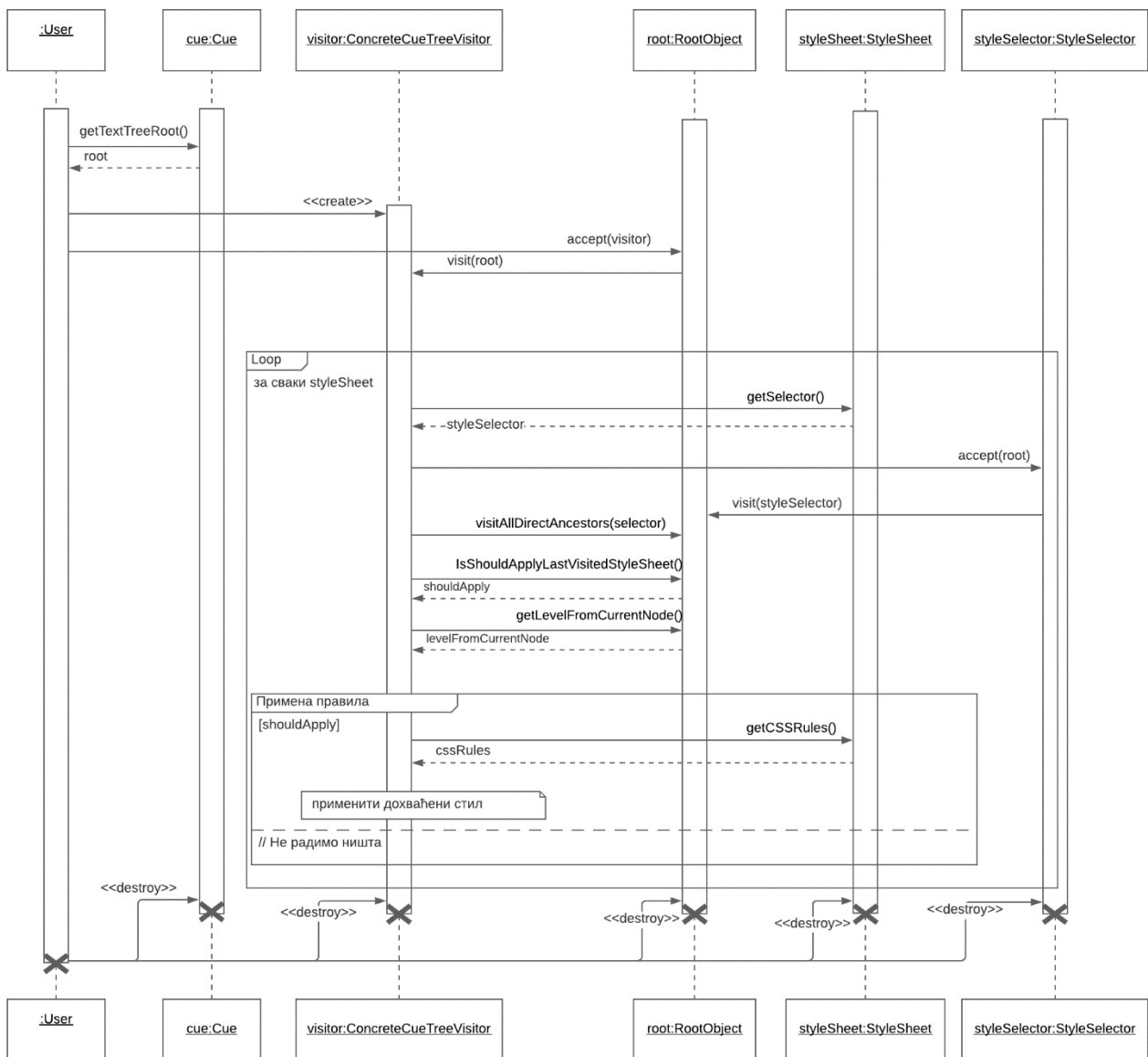
5.5. Обилазак стабла

Као што је речено, за парсирани текст превода формира се стабло. Како би се добио текст који треба приказати, потребно је обићи стабло у *preorder* [24] поретку: прво је потребно обићи чвор, затим његово лево подстабло, а онда се вратити на десно подстабло.

Како би се ово подржало, искоришћен је образац "Посетилац". Овим се дефинише интерфејс који има по једну методу за сваки тип чвора стабла. Када корисник позове над неким чвором прихватање посетиоца, биће позвана одговарајућа метода конкретног посетиоца која одговара типу чвора. Метода за прихватање посетиоца обилази само дати чвор, а уколико је потребно

обработити још неке чворове (претке или децу), потребно је да корисник који имплементира конкретног посетиоца посети и те друге чворове. Посета деце је могућа тако што се у методи посетиоца за обраду чвора позове метода за обраду деце над датим чвором. Методи се прослеђује објекат посетиоца. На овај начин корисник може позвати методу за посету над кореним чвором и тако обићи комплетно стабло. Како би се обишло стабло, потребно је у свакој методи посетиоца за конкретан чвор позвати и методу за посету деце. За подразумевано понашање је узето да се не обрађују деца, како би корисник који имплементира посетиоца имао већу флексибилност.

Идеја је да се током обиласка стабла за сваки чвор провери који све стилови треба да се примене на дати чвор. Доношење одлуке о томе да ли неки селектор треба применити је такође одговорност обрасца "Посетилац". Дефинише се још један интерфејс за образац посетилаца који ће служити за посету одговарајућег селектора. Слично као код чвора стабла текста, посетилац има по једну методу за сваки тип селектора. Како би се одлучило да ли неки селектор треба да се примени на неки чвор, потребно је позвати методу селектора за прихватање посетиоца. Како би било могуће да проследимо чвор као посетиоца, сваки чвор мора да имплементира интерфејс за обилазак селектора (интерфејс посетиоца). За услов примене селектора се гледа само прослеђени чвор, уколико се жели провера да ли се неки чвор на директној линији потомства до нашег чвора (пратећи показивач *parent* до корена) поклапа са тренутним селектором, потребно је позвати методу за посету директних предака. Метода прима селектор и над њим ће позвати прихватање посетиоца за сваки чвор до корена. Ово је корисно уколико се обилази стабло и потребно је знати све стилове који се примењују на чвор. Приликом примене стилова, мора се пазити који стил је дубље у стаблу, јер се прво примењују стилови који су ближи корену, па онда стилови на дубљем нивоу. Како би се ово омогућило, приликом посете предака води се рачуна који предак по реду се поклопио са задатим селектором. Корисник који примењује селекторе треба да дохвати овај број и да на основу њега одлучи којим стилима треба дати предност. Ово је битно само за поља стила која имају исто име. Слично, бафер који садржи стилове је потребно обилазити од почетка ка крају јер стилови треба да се примењују редоследом којим су наведени, односно већи приоритет имају стилови који су ближи крају бафера. Прво се гледа приоритет у стаблу (ко је ближи корену), а ако је тај приоритет исти онда се гледа позиција у баферу. Други начин подразумева памћење стилова које треба применити приликом уласка у методу за обилазак чвора. Стиливе је потребно уклонити после обиласка деце чвора, односно при напуштању методе за обилазак чвора. Осим стилова који се задају помоћу *CSS-a*, неки чворови стабла и сами за себе представљају одговарајући стил. Када се обилази неки чвор, зато је потребно додати и стил који представља тај чвор. Име чвора одговара стилу који се додаје. На пример, **BoldObject** представља подебљање. Такође, имена чворова одговарају ознакама у блоку за превод видеа (3.3.4.1). На следећој слици се налази дијаграм секвенце који показује пример употребе.



Дијаграм секвенце 4 Пример обиласка стабла

6. Додавање библиотеке у постојећи пројекат

У овом одељку ће бити објашњен начин на који се библиотека може додати у пројекат. Приликом превођења изворног кода, прави се фајл који представља дељену библиотеку, као и извршни фајл који је могуће покренути. Како би користила библиотека, потребно је направити библиотеку и додати је у пројекат. У наставку ће бити објашњен начин на који је то могуће урадити.

6.1. Клонирање пројекта са *GitHub-a*

У овом поглављу је објашњено клонирање пројекта [25] помоћи *SSH-a*. Како би се користио *SSH* [26], потребно је конфигурисати *GitHub* налог, као и машину која се користи. То је могуће урадити пратећи упутство са званичне странице *GitHub-a* [27]. Уколико је све конфигурисано, потребно је позиционирати се у директоријум у којем ће бити изворни код библиотеке. Након позиционирања потребно је укуцати следећу команду:

```
git clone git@github.com:duca9/WebVTT_Parser.git
```

Слика 30 Клонирање пројекта са *GitHub-a*

Издаз би требао да буде сличан издазу на следећој слици:

```
➤$ git clone git@github.com:duca9/WebVTT_Parser.git
Cloning into 'WebVTT_Parser' ...
remote: Enumerating objects: 1502, done.
remote: Counting objects: 100% (1502/1502), done.
remote: Compressing objects: 100% (812/812), done.
remote: Total 1502 (delta 777), reused 1330 (delta 631), pack-reused 0
Receiving objects: 100% (1502/1502), 7.68 MiB | 996.00 KiB/s, done.
Resolving deltas: 100% (777/777), done.
```

Слика 31 Издаз после команде

Ефекат ове команде је направљен директоријум са именом *WebVTT_Parser* у којем се налази изворни код библиотеке. Затим је потребно позиционирати се у направљени директоријум следећом командом: [28]

```
cd WebVTT_Parser/
```

Слика 32 Позиционирање у клонирани пројекат

Пошто пројекат користи библиотеку за *UTF-8* конверзију која се налази на *GitHub-u*, потребно је клонирати и ту библиотеку. *GitHub* води рачуна о модулима од којих зависи пројекат, тако да ће он клонирати модуле који су потребни. Довлачење потребних модула може се урадити на следећи начин. [29] Прво је потребно задати следећу команду:

```
git submodule init
```

Слика 33 Иницијализација модула од којих пројекат зависи

Изназ команде требало би да буде сличан излазу на следећој слици:

```
Submodule 'libwebvtt/lib/utfcpp' (git@github.com:nemtrif/utfcpp.git) registered for path 'libwebvtt/lib/utfcpp'
```

Слика 34 Излаз после команде за иницијализацију модула

Затим је потребно задати следећу команду:

```
git submodule update
```

Слика 35 Клонирање модула од којих пројекат зависи

После ове команде требало би да се клонира помоћни модул. Излаз би требао да буде сличан следећој слици:

```
Cloning into '/home/dusan/example/lib/WebVTT_Parser/libwebvtt/lib/utfcpp' ...  
Submodule path 'libwebvtt/lib/utfcpp': checked out 'b85efd66a76caccbe0c186b00cab34df1e4281fa'
```

Након ових корака скинут је изворни код библиотеке пројекта.

6.2. Прављење дељење библиотеке

Прво је потребно позиционирати се у директоријум у коме се налази фајл који се зове `makefile`. Уколико је изворни код библиотеке скинут пратећи претходни корак (6.1), потребно је задати следећу команду:

```
cd libwebvtt/
```

Слика 36 Позиционирање у директоријум дељене библиотеке

Фајл са именом `makefile` се налази у директоријуму `WebVTT_Parser /libwebvtt`, где је `WebVTT_Parser` директоријум који је добијен клонирањем пројекта у претходном кораку.

Прво је потребно очистити цео пројекат. То се може урадити користећи правило `clean` у фајлу са именом `make` и алат који се зове `make` [30]. Како би се извршило правило потребно је укуцати следећу команду.

```
make clean
```

Слика 37 Команда за чишћење пројекта

Затим је потребно покренути превођење, слично као за чишћење пројекта, може се користити `make` алат. Како би се превео пројекат потребно је укуцати следећу команду:

```
make
```

Слика 38 Команда за превођење пројекта

Овом командом ће се покренути превођење изворног кода библиотеке. [31]

Ефекат позива претходне команде је прављење новог директоријума `output_dir` у који ће се сместити фајл `libwebvtt.so`. Фајл који је направљен је у ствари дељена библиотека [32] која

може да се увезе у пројекат. Такође се прави и фајл `webvtt` који је извршни фајл и који представља пример коришћења. Извршни фајл прима један аргумент преко командне линије који представља апсолутну путању до фајла који треба да парсира. Резултат извршавања је извештај о томе шта је парсирано у фајлу. Изворни код показног примера се може наћи у фајлу `main.cpp`.

Осим директоријума, прави се и директоријум `build` у којем се налазе објектни фајлови који су коришћени за прављење библиотеке. Осим објектних фајлова, у директоријуму се налазе и фајлови који помажу `make` алату да прати промене у `hrr` фајловима, како би знао када треба опет превести одговарајући `cpp` фајл и направити нови објектни фајл, а затим и нови фајл библиотеке.

6.3. Покретање примера коришћења

Како би се покренуо пример коришћења, потребно је позиционирати се у директоријум `output_dir` и задати следећу команду:

```
./webvtt imeFajla.vtt
```

Слика 39 Покретање примера коришћења библиотеке

Аргумент `imeFajla.vtt` треба да буде апсолутна путања до фајла са *WebVTT* садржајем.

6.4. Додавање библиотеке приликом превођења

После прављења „*so*“ фајла потребно је рећи компајлеру да га користи. Како би компајлер знао где да тражи дељену библиотеку, потребно је приликом превођења изворног кода пројекта који користи библиотеку додати опцију за линкер `-L` и затим написати путању до директоријума у коме се налази дељена библиотека. Ради лакшег разумевања, у даљем тексту ће се путања где се налази изворни код библиотеке (путања где се налази и фајл са именом `makefile`) означавати `SHARED_LIB_HOME`. Пошто линкеру треба да зна где тражи *so* фајл, опција линкера је :

```
-LSHARED_LIB_HOME/output_dir
```

Слика 40 Опција линкера којом му се говори где да тражи библиотеке

У суштини, треба рећи линкеру где да тражи дељену библиотеку. Уколико се *so* фајл налази у неком другом директоријуму, онда треба ставити путању до тог директоријума. [31]

Како би се `hrr` фајлови користили помоћу релативних путања, а не апсолутних, потребно је рећи компајлеру где да их тражи. Када компајлер зна директоријум где треба да тражи фајлове, путање фајлова се задају релативно у односу на наведени директоријум (могу и апсолутно). Може се замислити да ће компајлер кренути из директоријума који му је дат као опција и одатле тражити путању која му је задата `include` директивом. Наравно, уколико желимо да користимо апсолутне путање, онда нема потребе за додавањем опције компајлеру. Како би се подесило где компајлер тражи фајлове, потребно је задати следеће две опције:


```
-ISHARED_LIB_HOME/include  
-ISHARED_LIB_HOME/lib/utfcpp/source
```

Слика 41 Опција компајлера којом му се говори где да тражи ".hrr" фајлове

Фајлови са екстензијом `hrr` су потребни компајлеру како би знао декларације функција и дефиниције класа. За употребу саме библиотеке нису потребни `сrr` фајлови. Они су потребни само како би се направила библиотека, односно `so` фајл. Кориснику библиотеке се препоручује да погледа структуру `hrr` фајлова како би знао шта треба да укључи како би користио одређени део библиотеке.

Приликом превођења пројекта који користи библиотеке, битан је редослед навођења фајлова компајлеру. Прво је потребно навести фајлове који користе библиотеку, а тек онда саму библиотеку. Ово је потребно због начина рада линкера, који повезује само библиотеке у којима је дефинисан тражени симбол. Ако прво ставимо библиотеку, ниједан симбол неће бити тражен и самим тим библиотека неће бити повезана.

Следећа опција коју треба задати линкеру је име библиотеке. За сваку наведену библиотеку `GCC` [33] подразумевано тражи динамичку библиотеку. Уколико је име библиотеке наведено са префиксом `l` након кога следи име без екстензије, на пример `lwebvtt`, онда ће линкер тражити библиотеку са именом `libwebvtt.so`. Управо зато је приликом прављења библиотека направљена библиотека са датим именом, а линкеру ће се проследити скраћена верзија имена. [34]

На крају би команда требало да изгледа као на следећој слици:

```
g++ imeFajla.cpp -LSHARED_LIB_HOME/output_dir -ISHARED_LIB_HOME/include -ISHARED_LIB_HOME/lib/utfcpp/source  
-lwebvtt -pthread -std=c++20 -o imelzlaza
```

Слика 42 Изглед команде за превођење пројекта који користи дељену библиотеку за парсирање `WebVTT` фајла

Фајл у којем се користи дељена библиотека за `WebVTT` парсер се зове `imeFajla.cpp` и он се преводи. Програм је потребно преводити преводиоцем за језик `C++20`, пошто се користи специјалан тип стринга специјално намењен за `UTF-8`, који је уведен тек од верзије `C++20`. Пошто дељена библиотека користи нити, потребно је рећи компајлеру да повеже и библиотеку која имплементира нити. Ово се омогућава опцијом `-pthread`. [35]

6.5. Покретање програма који је повезан са дељеном библиотеком.

Коришћењем дељене библиотеке смањујемо укупну величину извршног фајла. Такође, више програма може да користи исту дељену библиотеку и она ће бити само једном учитана у меморију, чиме се смањује потребна количина меморије уколико имамо више програма који се извршавају, а користе исту дељењу библиотеку. Као што је било потребно рећи линкеру где да тражи дељену библиотеку, исто тако треба рећи и динамичком линкеру [32] где ће тражити динамичке библиотеке када буде потребно да током извршавања прочита нови симбол. Ово је могуће урадити на више начина, од којих ће овде бити објашњен један. Наиме, постоји неколико подразумеваних начина помоћу којих динамички линкер тражи динамичке библиотеке. Један од тих начина је да претражи све локације које се налазе у променљивој

окружења (енгл. *environment variable*) `LD_LIBRARY_PATH`. Потребно је овој променљивој додати путању у којој се налази наша библиотека. Уколико се библиотека већ налази на некој локацији која се већ налази у променљивој, није потребно додати је опет. Пошто се променљива иницијализује на подразумевану вредност сваки пут када се покрене терминал, потребно је одрадити следеће кораке сваки пут кад покренемо терминал поново. [31]. Додавање се врши на следећи начин:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:SHARED_LIB_HOME/output_dir
```

Слика 43 Променљива којом се говори динамичком линкеру где да тражи библиотеке

На овај начин само додајемо нову путању, задржавајући старе путање, \$ [36]ознака уз име променљиве означава њену вредност, док ће се остатак линије надовезати на ту вредност. Знак : се ставља јер је то ознака за крај једне и почетак друге путање.

На овај начин имамо променљиву терминала која је недоступна за процесе које терминал покрене. Како бисмо омогућили да процеси које терминал направи имају приступ овој променљивој, потребно је да је извеземо. [31] То се ради на следећи начин:

```
export LD_LIBRARY_PATH
```

Слика 44 Омогућавање динамичком линкеру да види променљиву

На крају можемо покренути наш програм на следећи начин:

```
./imalzlaza [аргументи]
```

Слика 45 Покретање преведеног извршног фајла

6.6. Ограничења и препоруке

Ограничења и препоруке приликом коришћења библиотеке гласе:

- Библиотека је прављена за коришћење на систему *Linux*, тако да се може користити и даље развијати само на том систему, осим ако у будућим верзијама не буду подржани и други системи.
- Библиотека је развијана и тестирана помоћу компајлера *GCC*, па је препорука да се тај компајлер користи и у даљем развоју и коришћењу библиотеке.
- Уколико се користи дељена библиотека, препорука је да се користи исти компајлер који је коришћен и за прављење библиотеке, како би се дељена библиотека успешно повезала и учитала. Ово је потребно јер стандаром није дефинисано како треба да се праве и понашају дељене библиотеке, тако да различити компајлери као и различите верзије компајлера могу другачије да се понашају.

- Приликом коришћења библиотеке потребно је код преводити са ознаком компајлера `-std=c++20` како би се специфицирао стандард `C++` који треба да се користи. У библиотеци је коришћен овај стандард због новог типа за `UTF-8` стринг који је убачен у овој верзији стандарда.

7. Закључак

На основу свега реченог, може се рећи да је парсер *WebVTT* формата на језику *C++* успешно имплементиран. Током имплементације наишло се на неколико проблема. Највећи је био то што формат подржава део *CSS-a* који се може користити на разне начине. Пошто су библиотеке које парсирају *CSS* у вези са *HTML-ом*, написан је парсер за део *CSS-a*. Још један пројектни проблем везан за *CSS* су селектори и њихово упаривање. У *CSS-у* постоји много различитих врста селектора који се примењују на одговарајуће чворове нашег стабла са текстом. Могућност да немамо цео фајл одједном у баферу, него да се подаци довлаче преко интернета и убацују у бафер представљао је проблем за себе. Пошто је формат *UTF-8* енкодован, само парсирање фајла захтевало је посебан третман, јер један карактер није једнак једном бајту. То је решено пребацивањем у *UTF-32* како би се олакшало парсирање.

У наставку ће бити објашњене идеје за даљи развој библиотеке.

- Могућности за псеудо селекторе

Тренутно је у библиотеци подржано парсирање псеудо-селектора, међутим, како их има много, остављено је за касније да се имплементирају конкретни селектори, као и услов да ли се конкретни псеудо-селектор може применити на неки чвор стабла са текстом.

- Могућност за уједињене селекторе и више селектора у једном

Како је могуће комбиновати више селектора у један селектор, и како има доста различитих могућности на које се то може урадити, остављено је за касније да се имплементира логика да ли се уједињени селектор примењује на неки чвор стабла. Библиотека подржава парсирање ових врста селектора, али ће приликом парсирања бацити изузетак ако тип селектора није подржан.

- Неразликовање малих и великих слова

Пошто су неки делови *CSS-a* неосетљиви на велика и мала слова, а текст може бити *UTF-8* енкодован, и како није лако одрадити ово за сва могућа писма, остављено је за касније да се пронађе нека библиотека која то ради и да се убаци у пројекат.

- Поновно парсирање

Парсеру може да се постави бафер и да се онда започне парсирање.

- Поновно декодовање

Декодеру може да се постави бафер и да се онда започне декодирање.

- Лакша примена стилова

Треба имплементирати методу која прима чвор стабла и све стилове, а враћа стилове који треба да се примене на дати чвор стабла (узимајући у обзир и претке). Ова метода би разрешавала све приоритете за стилове који имају исто име. Тада ће корисник моћи у чвору *TextObject*, који представља сам текст, моћи да позове ову методу и вратиће му се сви стилови које треба позвати. Тренутно корисник мора сам да одради овај посао.

8. Литература

- [1] "WebVTT cue text parsing rules," 2019. [Online]. Available: <https://www.w3.org/TR/webvtt1/#cue-text-parsing-rules>.
- [2] "WebVTT: The Web Video Text Tracks Format," 2019. [Online]. Available: <https://www.w3.org/TR/webvtt1/#introduction>.
- [3] "Introducing JSON," [Online]. Available: <https://www.json.org/json-en.html>.
- [4] "<https://en.wikipedia.org/wiki/HTML5>," 2021. [Online].
- [5] "Can I use," [Online]. Available: <https://caniuse.com/webvtt>.
- [6] "ETSI," 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/103200_103299/103285/01.02.01_60/ts_103285v010201p.pdf.
- [7] "Cascading Style Sheets," [Online]. Available: <https://www.w3.org/Style/CSS/>.
- [8] "UTF-8," 2020. [Online]. Available: <https://sr.wikipedia.org/wiki/UTF-8>.
- [9] "WebVTT (Web Video Text Tracks)," [Online]. Available: https://www.speechpad.com/captions/webvtt#toc_3.
- [10] "XML," 2021. [Online]. Available: <https://en.wikipedia.org/wiki/XML>.
- [11] "TTML (Timed Text Markup Language)," 2021. [Online]. Available: <https://www.speechpad.com/captions/ttml>.
- [12] "Content delivery network," 2021. [Online]. Available: https://en.wikipedia.org/wiki/Content_delivery_network.
- [13] "The MPEG-DASH Standard," [Online]. Available: https://www.bogotobogo.com/VideoStreaming/images/mpeg_dash/T_MM1_TheMPEGDASHStandard.pdf.
- [14] "Named character references," 2021. [Online]. Available: <https://html.spec.whatwg.org/multipage/named-characters.html>.
- [15] "Introduction to the DOM," 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction.
- [16] "Pseudo-classes," 2021. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes>.
- [17] "Pseudo-elements," 2021. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements>.
- [18] "WebVTT API," [Online]. Available: <https://www.w3.org/TR/webvtt1/#api>.
- [19] "Composite," [Online]. Available: <https://refactoring.guru/design-patterns/composite>.
- [20] "State Machine Design pattern — Part 2: State Pattern vs. State Machine," 2019. [Online]. Available: <https://medium.datadriveninvestor.com/state-machine-design-pattern-part-2-state-pattern-vs-state-machine-3010dd0fcf28>.
- [21] "Flyweight," 2021. [Online]. Available: <https://refactoring.guru/design-patterns/flyweight>.
- [22] "Cleaner Code with Static Factory Methods," 2017. [Online]. Available: <https://stackify.com/static-factory-methods/>.
- [23] "UTF-32," 2021. [Online]. Available: <https://en.wikipedia.org/wiki/UTF-32>.

- [24] "Preorder Tree Traversal – Iterative and Recursive," [Online]. Available: <https://www.techiedelight.com/preorder-tree-traversal-iterative-recursive/>.
- [25] "Cloning a repository," [Online]. Available: <https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository>.
- [26] "How Does SSH Work," 2021. [Online]. Available: <https://www.hostinger.com/tutorials/ssh-tutorial-how-does-ssh-work>.
- [27] "Connecting to GitHub with SSH," 2021. [Online]. Available: <https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh>.
- [28] "Cd Command in Linux (Change Directory)," 2021. [Online]. Available: <https://linuxize.com/post/linux-cd-command/>.
- [29] "Git Submodules basic explanation," [Online]. Available: <https://gist.github.com/gitaarik/8735255>.
- [30] "Linux 'make' Command Explained With Examples," 2021. [Online]. Available: <https://linuxide.com/linux-make-command-examples/>.
- [31] "How to create shared library (.SO) in C++ (G++)?," [Online]. Available: <https://iq.opengenus.org/create-shared-library-in-cpp/>.
- [32] "Dynamic Linking," 2021. [Online]. Available: https://en.wikipedia.org/wiki/Dynamic_linker#:~:text=In%20computing%2C%20a%20dynamic%20linker,jump%20tables%20and%20relocating%20pointers..
- [33] "GCC, the GNU Compiler Collection," 2021. [Online]. Available: <https://gcc.gnu.org/>.
- [34] "gcc -I option flag," [Online]. Available: <https://www.rapidtables.com/code/linux/gcc/gcc-i.html>.
- [35] "Compiling C program with pthread.h library in Linux," 2018. [Online]. Available: <https://www.includehelp.com/c-programming-questions/compiling-program-with-pthread-library-linux.aspx>.
- [36] "Expansion," 2021. [Online]. Available: https://linuxcommand.org/lc3_lts0080.php.
- [37] "https://en.wikipedia.org/wiki/World_Wide_Web," 2021. [Online].
- [38] "The Standard," 2021. [Online]. Available: <https://isocpp.org/std/the-standard>.

Списак слика

Дијаграм класа 1 Бафер	20
Дијаграм класа 2 Структура парсера.....	21
Дијаграм класа 3 Структура излаза парсера.....	22
Дијаграм класа 4 Структура стабла за текст превода.....	22
Дијаграм класа 5 Парсирање превода видеа	24
Дијаграм класа 6 Структура парсера блока са стилем	25
Дијаграм класа 7 Структура потребна за проверу поклапања селектора.....	26
Дијаграм класа 8 Структура потребна за обилазак стабла превода.....	27
Дијаграм секвенце 1 Пример употребе бафера	28
Дијаграм секвенце 2 Пример употребе декодера.....	29
Дијаграм секвенце 3 Пример употребе парсера.....	30
Дијаграм секвенце 4 Пример обиласка стабла	32
Слика 1 Пример фајла у SRT формату.....	4
Слика 2 Пример употребе TTML формата	4
Слика 3 Инфраструктура MPEG-DASH клијента [13]	5
Слика 4 Изглед MPD- а [13].....	6
Слика 5 Пример структуре WebVTT формата	7
Слика 6 Пример метаподатака заглавља [9].....	8
Слика 7 Приказ основних делова блока са преводом.....	8
Слика 8 Формат временске ознаке	9
Слика 9 Формат коришћења ознака	12
Слика 10 Пример употребе гласовне ознаке	12
Слика 11 Пример употребе ознаке за курзив	12
Слика 12 Пример употребе тага за подебљане текста.....	13
Слика 13 Пример употребе ознаке за подвлачење текста.....	13
Слика 14 Пример употребе временске ознаке	13
Слика 15 Пример употребе ознаке за језик	13
Слика 16 Пример употребе ознаке за класу	14
Слика 17 Пример употребе тага за горњи део текста.....	14
Слика 18 Ефекат ознаке за текст изнад текста.....	14
Слика 19 Пример употребе хексадецималне референце.....	14
Слика 20 Пример употребе децималне референце	15
Слика 21 Пример подешавања региона	15
Слика 22 Објашњење подешавања regionanchor и viewportanchor [2].....	16
Слика 23 Пример стилизовања региона.....	17
Слика 24 Стил који се примењује на све блокове са текстом	17
Слика 25 Пример стилизовања блока са текстом помоћу идентификатора.....	17
Слика 26 Пример стилизовања блока са текстом помоћу селектора по типу ознаке	18
Слика 27 Пример стилизовања блока са текстом помоћу селектора класе	18
Слика 28 Пример стилизовања блока са текстом помоћу селектора са атрибутом	19
Слика 29 Пример употребе future и past псеудо селектора.....	19
Слика 30 Клонирање пројекта са GitHub-а.....	33

Слика 31 Излаз после команде	33
Слика 32 Позиционирање у клонирани пројекат	33
Слика 33 Иницијализација модула од којих пројекат зависи	33
Слика 34 Излаз после команде за иницијализацију модула	34
Слика 35 Клонирање модула од којих пројекат зависи	34
Слика 36 Позиционирање у директоријум дељене библиотеке	34
Слика 37 Команда за чишћење пројекта	34
Слика 38 Команда за превођење пројекта	34
Слика 39 Покретање примера коришћења библиотеке	35
Слика 40 Опција линкера којом му се говори где да тражи библиотеке	35
Слика 41 Опција компајлера којом му се говори где да тражи ".hrr" фајлове	36
Слика 42 Изглед команде за превођење пројекта који користи дељену библиотеку за парсирање WebVTT фајла	36
Слика 43 Променљива којом се говори динамичком линкеру где да тражи библиотеке	37
Слика 44 Омогућавање динамичком линкеру да види промењиву	37
Слика 45 Покретање преведеног извршног фајла	37

Списак табела

Табела 1 Ефекат поравњања у зависности од правца текста [9]	9
Табела 2 Ефекат поравњања у зависности од правца текста, поравњања текста и позиције [9]	10
Табела 3 Ефекат подешавања линије у зависности од правца и поравњања [9]	11