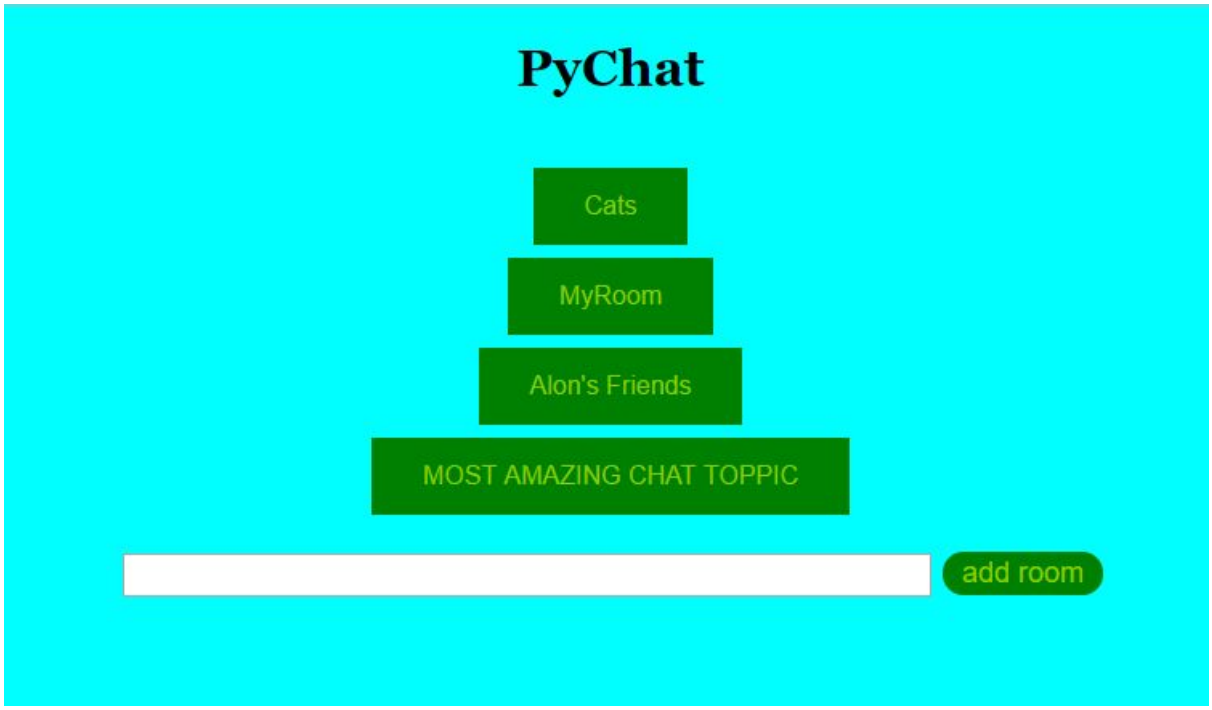
פרוייקט סייבר במסגרת תוכנית גבהים

Python based HTTP chat



מקסימיליאן מטיאש
318828761
תיכון ליאו-באק
הגנת סייבר

מנחים

אלון בר-לב

שרית לולב

תוכן עניינים

תוכן עניינים **1**

מבוא **2**

רקע תאורתי **3**

תכנות אסינכרוני 3

פרוטוקול HTTP[RFC-2616] 3

ערוץ תקשורת סמוי 4

טכנולוגיה 4

ארכיטקטורה **6**

ישויות 7

תהליכים 7

מבני נתונים **8**

Block Diagram **10**

פרוטוקול תקשורת **11**

שירותים 11

שירותי צ'אט 11

get-rooms/ 11

get-messages/ 11

add-room: 12

שירותי HTML 12

chat/ 12

register/ 12

rooms/ 12

מכונת מצבים **13**

התקנה ותפעול **16**

התקנת השרת 16

גישת לקוחות 17

כניסה והוספת חדרים 18

מגבלות ידועות **19**

חלק אישי **19**

תוכניות עתידיות **19**

מבוא

הפרויקט הוא שרת HTTP אסינכרוני המספק שירותי תקשורת ללקוחות המתחברים אליו. הלקוחות מתקשרים זה עם זה באמצעות התחברות ושליחת הודעות לתוך חדרי צ'אט.
הלקוחות המתחברים לשרת תחילה נרשמים ובוחרים את שמם, השרת יודע לזהות אותם באמצעות cookie ייחודי לכל משתמש שנוצר בתהליך ההרשמה.
לאחר מכן כל לקוח מקבל דף HTML ובו מוצגים כל חדרי הצ'אט הקיימים בשרת. הלקוח יכול ליצור חדר חדש או להיכנס לתוך חדר קיים. כשהלקוח נכנס לחדר הוא מקבל דף HTML בכרטיסייה חדשה שבו מוצג החדר, ההודעות שנשלחו בו ורשימת המשתמשים הנמצאים בו כרגע.
הלקוח מתקשר עם השרת באמצעות XML דינמי שממומש דרך אפליקציית javascript על מנת שהחדר יתעדכן בזמן אמת ללא צורך בטעינה מחדש של הדף.
כל לקוח יכול להיות מחובר ולשלוח הודעות למספר חדרים בו זמנית. השרת יכול לתמוך בו זמנית בכמות גדולה של לקוחות.
המטרה היא לספק שירותי תקשורת זמינים וקלים לשימוש זאת כיוון שלא נחוצה התקנה כלל של תוכנות חיצוניות מצד המשתמש, ממשק המשתמש כולו דרך הדפדפן.
הפרויקט רץ ללא תלות במערכת הפעלה ודורש python 2.7 מותקן על מחשב השרת.

רקע תאורתי

תכנות אסינכרוני

טיפול באירועים הקורים תוך כדי זרם התוכנה מבלי לחסום את אותו הזרם בכדי לחכות לתוצאות. בפרט, באמצעות תכנות אסינכרוני ניתן לטפל במספר ערוצי קלט/פלט מבלי להשתמש ב-multithreading. הדבר אפשרי כיוון שלא מחכים בשום רגע עבור קלט או פלט מערוץ, זאת אומרת ששום ערוץ לא חוסם ערוץ אחר.
Asynchronous I/O ממומש באמצעות אובייקט poller. עוברים בלולאה על רשימת אובייקטים של קלט/פלט וקוראים ל-select system call עבור windows או poll system call עבור unix. קריאת המערכת מבצעת sleep עד שאחד מהבאים קורה: אירוע קלט/פלט ב-fd, timeout או signal. כשהפולר "מתעורר" הוא מחזיר את ה-fd ואת האירועים שהתרחשו בו. בודקים אילו אירועים התרחשו באמצעות or בין מה שהוחזר לבין המספר המתאים לאירוע ובהתאם לזאת מפעילים את המטודות המתאימות של אותו אובייקט עבור על אירוע (OnError, OnWrite, OnRead).

פרוטוקול HTTP[RFC-2616]

פרוטוקול תקשורת מבוסס טקסט הנמצא בשכבת היישום של מודל OSI ונועד להעברת hypermedia כלומר, שילוב של קובצי תמונה, קול וטקסט. רוב הרשת הכלל עולמית מבוססת על פרוטוקול זה.
בקשת HTTP מורכבת ממספר חלקים, כשהחלק החשוב בהם הוא השורה הראשונה של הבקשה, בה מופיעה שיטת הבקשה, הכתובת של מה שמבוקש (כתובת זו היא לרוב יחסית ולא מלאה מה שמחייב כותרת המכילה את שם המתחם של האתר אליו מיועדת הבקשה בחלק הבא של הבקשה) וגירסת הפרוטוקול. בסוף כל שורה בבקשה (וכן בתשובה) מופיעים התווים \r\n שמשמעותם שורה חדשה.
החלק הבא בבקשה הוא שדות כותרת (headers-הדרים) שמכילים מידע הקשור לבקשה, ללקוח או לתוכן הבקשה (אורכו וסוגו).
כאמור, החלק האחרון הוא תוכן הבקשה והוא אובייקט שהלקוח רוצה להעביר לשרת (לדוגמה להעלות תמונה).
חלק זה נפרד משאר הבקשה בעזרת שורה ריקה (שורה המכילה רק \r\n) וגם מסתיים בשורה ריקה.
שיטות נפוצות הן:
GET- מיועדת לקבלת אובייקט שנמצא על השרת, חסרת תוכן.
POST- מכילה תוכן, משמשת לשליחת מחרוזות לשרת, לרוב לצורך עיבוד.
תשובת HTTP זהה במבנה לבקשה מלבד לשורה הראשונה המתחילה בגירסת הפרוטוקול לאחריה מופיע קוד מצב התוצאה עם הסבר טקסטואלי קצר.
בשורות הבאות, כמו בבקשה, מופיעות כותרות המכילות מידע על השרת והתשובה ובסוף ישנו תוכן התשובה שתלוי בשיטת הבקשה ובקוד המצב.
קודי המצב מחולקים ל-5 מדורים:
1XX- ההודעה מכילה מידע בלבד.
2XX - הבקשה ששלח הלקוח בוצעה בהצלחה על ידי השרת, והתשובה מכילה את מה שהשרת נתבקש לשלוח.
3XX - הבקשה פוענחה בהצלחה, אך מסיבות כלשהן התשובה אינה כוללת את האובייקט המבוקש (למשל, משום שהעמוד מפנה אוטומטית לעמוד אחר).
4XX - נמצאה שגיאה כלשהי בבקשה עצמה.
5XX- התרחשה שגיאה פנימית בשרת.

ערוץ תקשורת סמוי

ערוץ סמוי הוא ערוץ תקשורת שניתן לנצל על מנת להעביר מידע באופן שפוגע במדיניות האבטחה של המערכת. ערוצים סמויים מהווים סכנה ממשית לאבטחת מידע שכן בשל אופיים הם מעבירים את המידע בצורה שאינה קלה להבחנה ולכן יכולים לפעול במשך זמן רב מבלי להתגלות וכתוצאה לפגוע קשות בסודיות מידע.
הערוץ הסמוי הוא ערוץ תקשורת פרזיטי שמטרתו להעביר מידע באופן שלא מפר את מדיניות האבטחה של האפליקציה. קשה להבחין בו כיוון שלא קיים קשר בין המידע הגלוי שעין שעובר בו לבין המידע האמיתי שעובר בו. ניתן להשתמש בפרוטוקול הצ'אט בכדי לשלוח מספר הודעות סתמיות בכל פעם כיוון שהפרוטוקול אוגר הודעות ושולח את כולן לאחר זמן קצר, לכן לא יחשדו אם יגיעו לכל היותר 4 הודעות ברצף (מהירות הקלדה סבירה של בן אדם). נתכנן ערוץ סמוי בצורת אפלקציית javascrip זדונית שבונה מעל לאפליקציית הצ'אט הרגילה. באפליקציה החדשה קיים שדה קלט חדש בנוסף לשדה הרגיל המיועד להודעות שנשלחות לחדר. בשדה החדש תמצא ההודעה שאותה אנו רוצים להעביר ונקודד אותה באמצעות הודעות סתמיות בשדה הרגיל. בשדה הרגיל נשים הודעה ממש ארוכה על מנת לשרת את פרוטוקול הקידוד. פרוטוקול הקידוד יחלק את ההודעה הסתמית וכל כמות זמן יישלח לפחות אחד ולכל היותר 4 תת הודעות בפרק זמן קצר כשהקידוד ממפה ביטים לכמות הודעות באופן הבא:

- הודעה 1: 00
- 2 הודעות: 01
- 3 הודעות: 10
- 4 הודעות: 11

כך נוכל לקודד באמצעות 6 ביטים 64 תווים אפשריים שזה מספיק לכל האותיות באנגלית (גדולות וקטנות), הספרות 0-9, סימן שאלה וסימן קריאה.
בעלי האפליקציה יוכלו לראות בחלון נפרד את ההודעות הנשלחות בערוץ הסמוי.
כיצד תזהה האפליקציה משתפי פעולה ?
כל פרק זמן האפליקציה תשלח הודעה מקודדת שנקבעה מראש דרך הערוץ הסמוי ותתחיל לפענח לפי פרוטוקול הקידוד את כל ההודעות שנשלחות בחדר. אם היא תזהה שמשתמש שלח הודעה זו היא תסמן אותו כמשתף פעולה. בסיום שלב זה היא תנסה לפענח רק הודעות שמקורן הוא משתף פעולה.

טכנולוגיה

פרוטוקול TCP:
פרוטוקול תקשורת בשכבת התעבורה המאפשר העברת נתונים באופן אמין וללא איבוד מידע.

שרת:
תוכנת מחשב המספקת שירותים לתוכנות לקוח באמצעות תקשורת רשתית.

XML:
תקן לייצוג נתונים המקל על החלפת נתונים בין מערכות שונות ומהווה דרך נוחה לשמור על נתונים דינמיים.

HTML:
שפת תגיות לתצוגה של דפי אינטרנט. באמצעותו המשתמש רואה ויזואלית את המידע שהשרת מחזיר דרך הדפדפן.

CSS:
פורמט לעיצוב דפי אינטרנט.

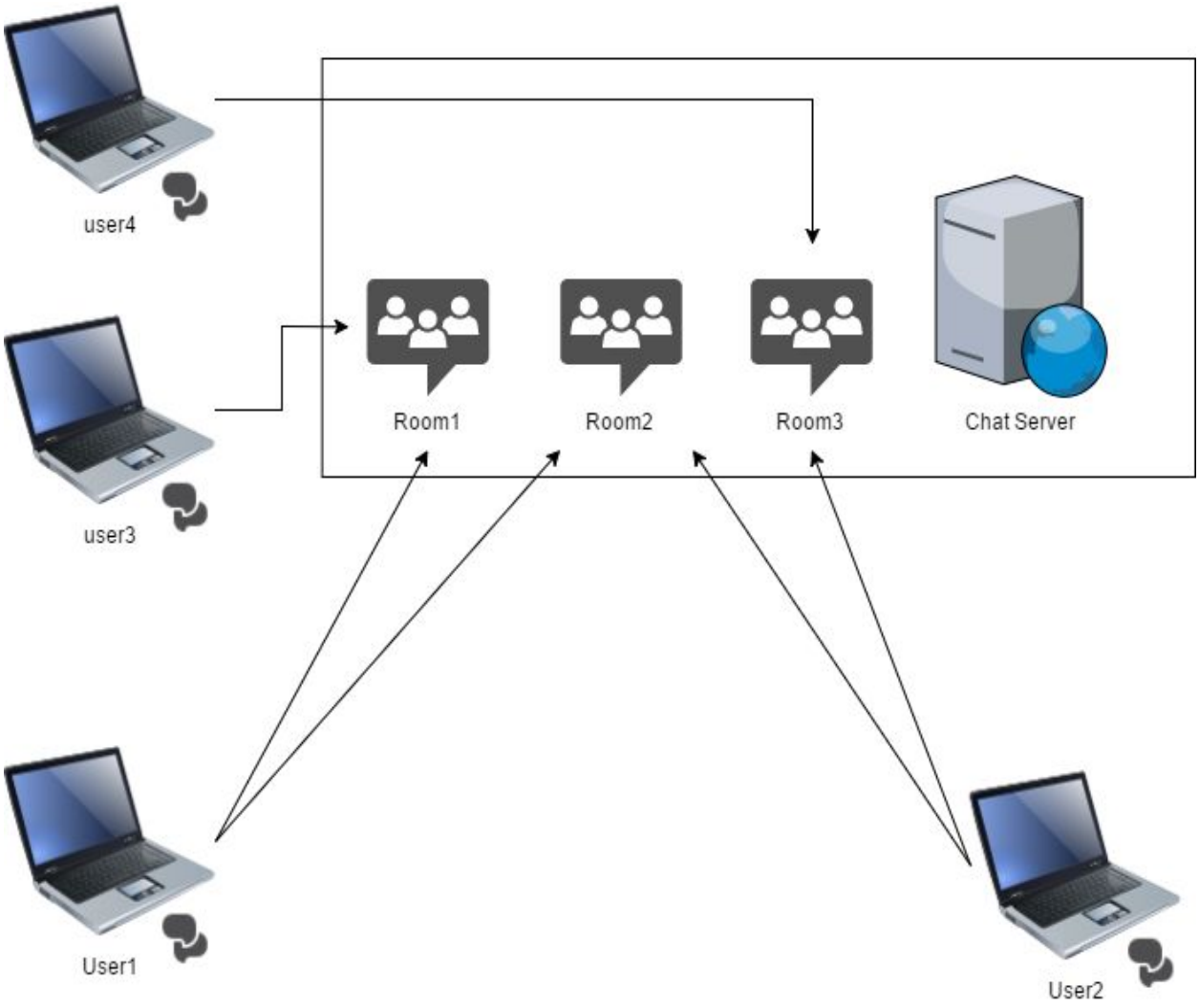
JavaScript:
שפה מונחית עצמים המותאמת להרצה בדפי אינטרנט.

Python 2.7:
שפת תכנות דינמיט נפוצה שבה כתוב הפרוייקט.

תכנות מונחה עצמים:
תכנות המבוסס על חלוקת התוכנית לעצמים בעלי תכונות ומטודות משלהם על מנת לחלק מערכת גדולה לרכיבים קטנים שביחד מהווים את כל התוכנית.

Async XMLHTTPRequest:
בקשות דינמיות של לקוח משרת המשתמשות ב-XML על מנת להעביר נתונים לשרת ומקבלות ממנו נתינים חדשים באותו אופן. מטרת שיטה זו היא למנוע טעינה מחדש של דף ולגרום לו להתעדכן באופן אוטומטי.
תרשימי זרימה:
תרשימים שנועדו לתאר יחסים בין יישויות בכדי להקל ולהבהיר את אופן פעולתן של מערכות מורכבות.

ארכיטקטורה



ישויות

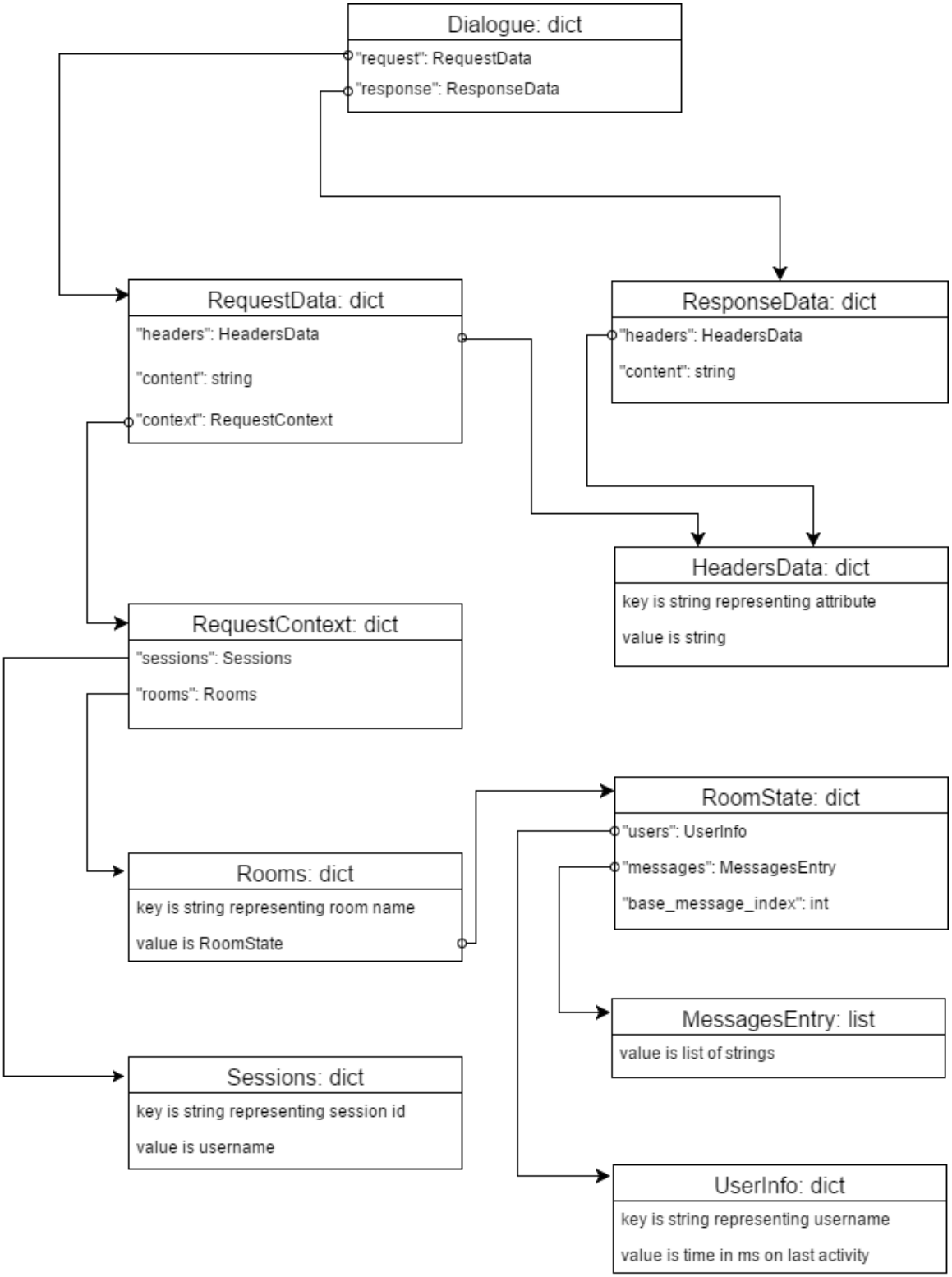
ישות	תיאור
server	השרת מאזין על פורט קבוע ומתחברים אליו לקוחות. השרת מקבל בקשות ומחזיר תשובות לכל בקשה.
room	חדר שיחה אליו מחוברים משתמשים. המשתמשים שולחים לחדר הודעות ומקבלים הודעות של משתמשים אחרים בחדר זה.
user	המשתמש מתחבר לשרת באמצעות דפדפן. הלקוח ראשית נרשם ובוחר את שמו ולאחר מכן יכול ליצור ולהתחבר לחדרים בהם הוא שולח ומקבל הודעות.

תהליכים

תהליך	תיאור
משתמש-חדר	המשתמש שולח הודעה לחדר, ההודעה מאוחסנת מקומית לזמן קצר.
חדר-שרת	ההודעות המאוחסנות בחדר נשלחות לשרת ומאוחסנות שם לזמן ארוך.
שרת-חדר	השרת מחזיר את ההודעות האחרונות שהמשתמש עוד לא ראה.
חדר-משתמש	ההודעות מגיעות לדפדפן של הלקוח והוא רואה אותן מוצגות ב-HTML.

דוגמה:
המשתמש user1 מחובר לחדרים room1, room2. בחדר room1 הוא מתקשר עם user3, בחדר room2 הוא מתקשר עם user2.

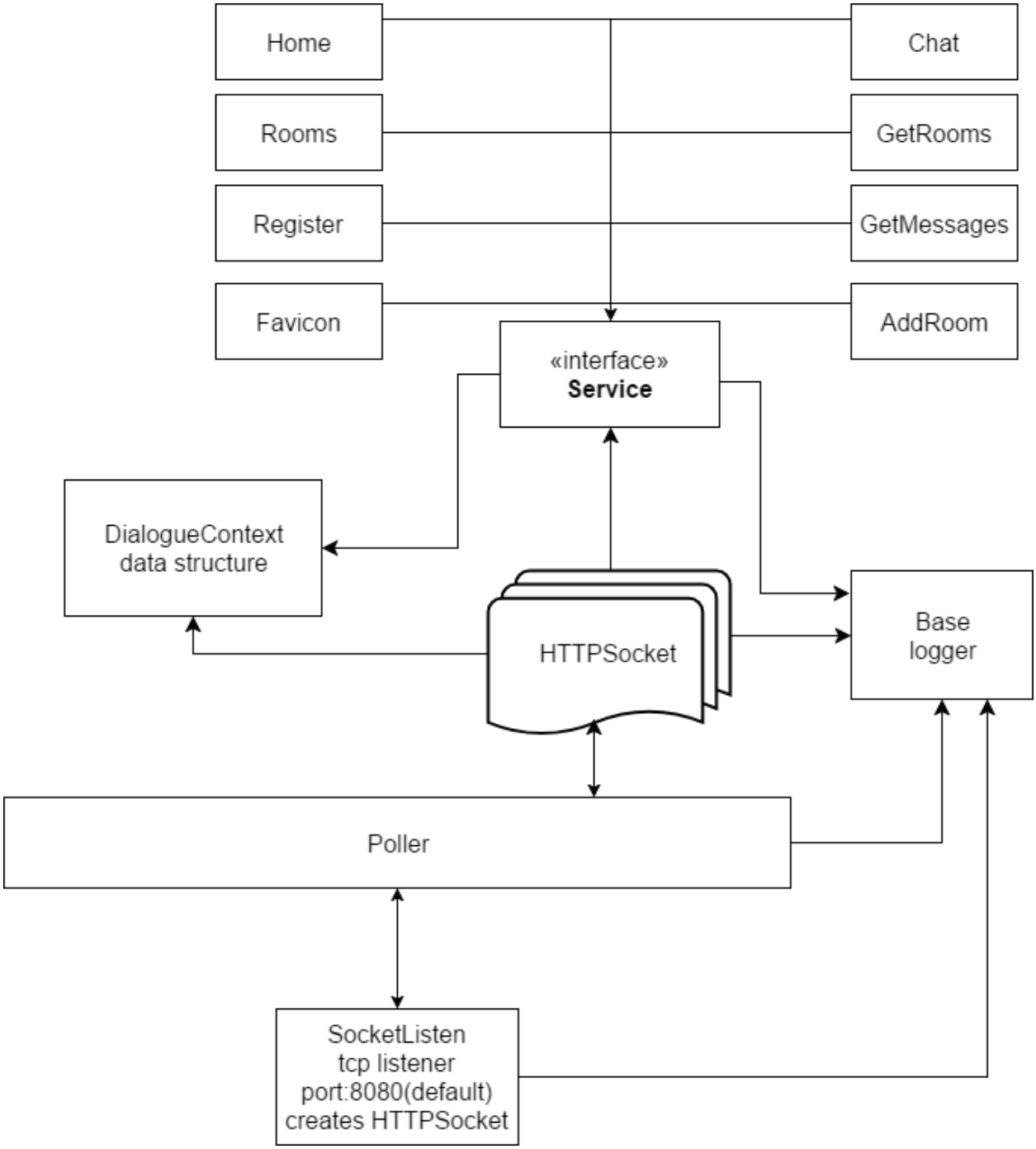
מבני נתונים


Dialogue: dict
"request": RequestData
"response": ResponseData
RequestData: dict
"headers": HeadersData
"content": string
"context": RequestContext
ResponseData: dict
"headers": HeadersData
"content": string
HeadersData: dict
key is string representing attribute
value is string
RequestContext: dict
"sessions": Sessions
"rooms": Rooms
RoomState: dict
"users": UserInfo
"messages": MessagesEntry
"base_message_index": int
Rooms: dict
key is string representing room name
value is RoomState
MessagesEntry: list
value is list of strings
Sessions: dict
key is string representing session id
value is username
UserInfo: dict
key is string representing username
value is time in ms on last activity

אלו הם מבני הנתונים בהם השרת משתמש בשירותיו על מנת להעביר מידע בין ישויות. כל מבנה מוגדר כ-שם:טיפוס. במבנים שאורך תוכנם קבוע מפורט התוכן. במבנים שאורך תוכנם משתנה מצוין הטיפוס של האובייקטים בתוך המבנה.

Block Diagram

מתוארים האובייקטים הקיימים בתוכנית והיחסים ביניהם.



פרוטוקול תקשורת

פרוטוקול התקשורת של השרת על לקוחותיו הוא [RFC-2616] HTTP ותוכן הבקשות והתשובות הוא [XML w3c]. שירותים מסויימים מקבלים תשובות בצורת קבצי html.

שירותים

שירותי צ'אט

/get-rooms

תיאור: בקשה לקבל רשימת חדרי צ'אט קיימים.
תוכן בקשה: ללא.
תוכן תשובה: XML המכיל את החדרים. דוגמה:

```
<root>
	<room name="room1"/>
	<room name="another room"/>
</root>
```

/get-messages

תיאור: אינטראקציית הצ'אט.
תוכן בקשה: XML המכיל את שם החדר, מצב ההודעות בחדר (ההודעות האחרונות שנראו) ו-0 או יותר הודעות. דוגמה:

```
<root>
	<!-- Last revision seen -->
	<fetch id="15">
	<room name="Stuff and things"/>
	<messages>
		<message text="hello"/>
	</messages>
</root>
```

תוכן תשובה: XML המכיל את ההודעות החדשות ורשימה של המשתמשים המחוברים לחדר ומעדכן את האינדקס וגרסה אחרונה של החדר. מספר הודעות יכולות להיחשב כגרסה אחת. דוגמה:

```
<root>
	<messages>
		<message text="Hi"/>
		<message text="hello"/>
		<message text="world"/>
	</messages>
	<!-- New revision -->
	<id revision="18"/>
	<users>
		<user name="Fred"/>
		<user name="baloon123"/>
	</users>
</root>
```

add-room:

תיאור: יצירת חדר חדש.
תוכן בקשה: XML עם שם החדר. דוגמה:

```
<root>
       <room name="MyRoom"/>
</root>
```

תוכן תשובה: ללא.

שירותי HTML

/chat

תיאור: בקשה לקבל חדר צ'אט.
פרמטרים: room שם החדר. (query string)
תוכן בקשה: ללא.
תוכן תשובה: הקובץ chat.html

/register

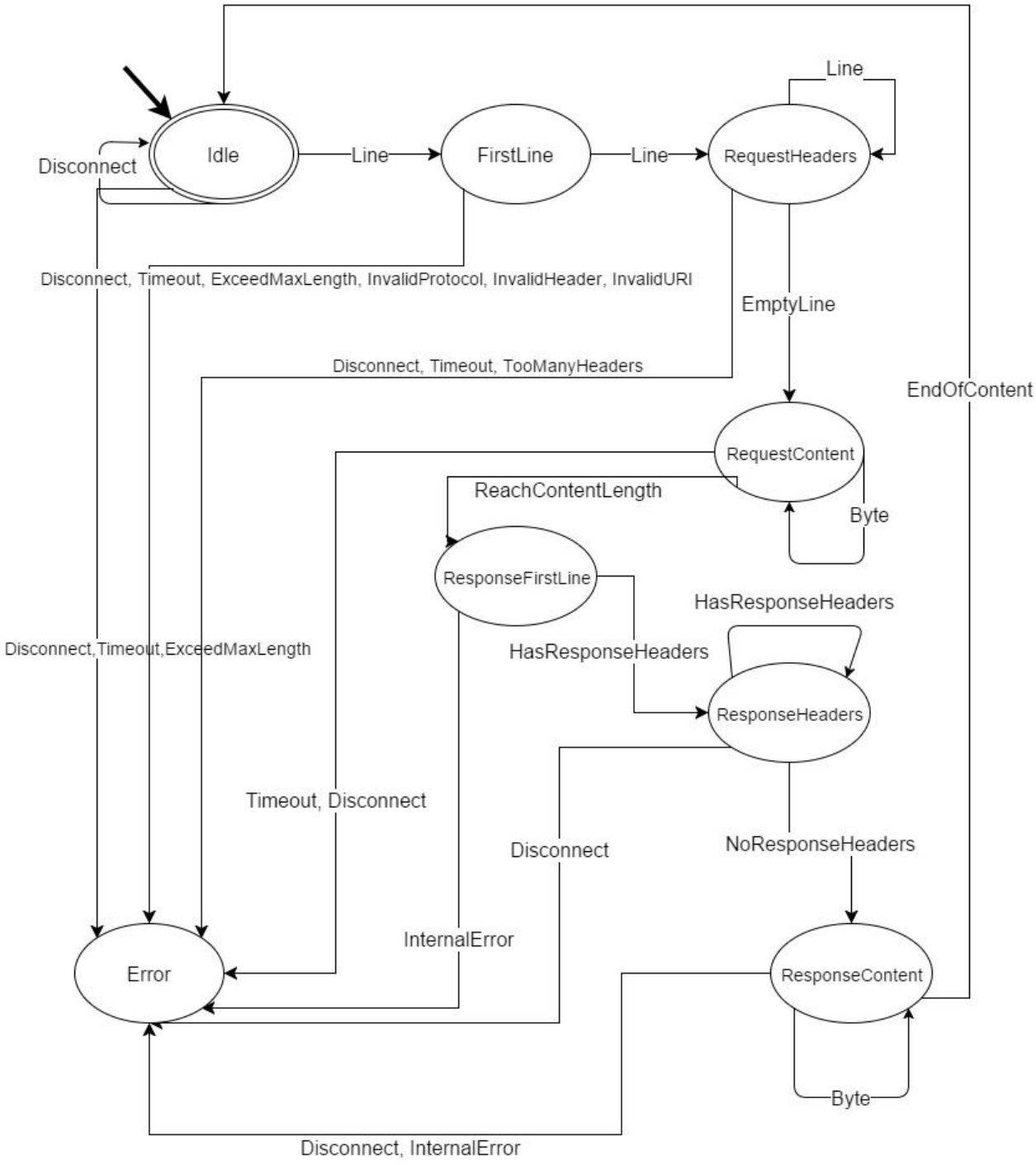
תיאור: רישום המשתמש בשרת וכניסה לרשימת החדרים.
פרמטרים: name שם המשתמש. (query string)
תוכן בקשה: ללא.
תוכן תשובה: ללא. (רדירקט לשירות room/ דרך הדר).

/rooms

תיאור: בקשה לקבל את הדף המציג את רשימת החדרים.
תוכן בקשה: ללא.
תוכן תשובה: הקובץ rooms.html

מכונת מצבים

מכונת המצבים מתארת תקשורת באמצעות פרוטוקול HTTP. בכל מצב מופעלת לוגיקה מתאימה ומבוצעות קריאות למטודים של השירות המתאימים למצב.



מצבים

שם	תיאור	הפלט עבור כל קלט	אירוע באיטרציה (אם נקרא שוב ושוב)
Idle	מצב התחלתי, לפני תקשורת.	Disconnect - ניתוק הלקוח. EndOfContent - שליחת התשובה וניתוק הלקוח.	
FirstLine	פארסינג ובדיקה קפדנית של השורה הראשונה.	Line - פריסת השורה הראשונה ושמירתה במבנה נתונים באם היא חוקית. קביעת ההדרים הרלוונטיים שנרצה לשמור ע"י הוספת שמם למבנה התנונים.	
RequestHeaders	קבלת הדרים של הבקשה.	Line - פריסת כל שורה ושמירתה במבנה נתונים באם היא חוקית ורלוונטית לשירות.	
RequestContent	קבלת תוכן בקשה.	EmptyLine - תחילת קבלת תוכן הבקשה. Byte - הוספת המידע למבנה נתונים תחת תוכן הבקשה.	request_content += byte
ResponseFirstLine	הכנת תשובה. לרוב ביצוע פעולות כמו פתיחת קבצים רלונטיים וכד'.	ReachContentLength - הכנות מקדימות לטיפול בבקשה כמו פתיחת קבצים.	
ResponseHeaders	הכנת הדרים של התשובה.	HasResponseHeaders - כתיבת ההדרים של התשובה במבנה נתונים.	headers[title] = data
ResponseContent	תוכן תשובה.	NoResponseHeaders - תחילת כתיבת תוכן התשובה. Byte - כתיבת תוכן התשובה למבנה נתונים.	response_content += byte
Error	שגיאה, ניתוק לקוח.	כל קלט - ניתוק הלקוח.	

אירועי קלט

אירוע קלט	תיאור
Line	שורה של תווים.
EmptyLine	שורה ריקה מהצורה \r\n.
Byte	בייט בינארי של מידע.
ReachContentLength	שאורך תוכן הבקשה הגיע לאורכו כמצוין בהדר Content-Length.
HasResponseHeaders	השירות המטפל בבקשה סיים הכנות מקדימות ומוכן לשים הדרים.
NoResponseHeaders	השירות סיים לכתוב את כל ההדרים הרלוונטיים בתשובה.
Disconnect	החיבור נסגר ע"י המשתמש.
Timeout	נפסק הזמן שהוקצב לחקות לאירוע.
ExceedMaxLength	אורך הבקשה גדול מדי.
InvalidProtocol	פרוטוקול הבקשה הוא לא HTTP.
InvalidHeader	ההדר שהתקבל לא חוקי.
InvalidURI	אין שירות שיכול לטפל ב-URI של הבקשה.
TooManyHeaders	כמות ההדרים גדולה מדי.
InternalError	שגיאה פנימית של השרת.

אירועי פלט

Byte	בייט בינארי של מידע.
EndOfContent	השירות סיים לכתוב את תוכן התשובה.
Disconnect	החיבור נסגר ע"י המשתמש.
Timeout	נפסק הזמן שהוקצב לחקות לאירוע.
ExceedMaxLength	אורך הבקשה גדול מדי.

התקנה ותפעול

התקנת השרת

תרם ההתקנה יש לוודא שמותקן python 2.7 על המחשב. אם לא יש להוריד מהקישור:
https://www.python.org/downloads/

בהתקנה, יש לוודא כי האפשרות add python to path סומנה.

בסיום ההורדה של השרת יש לחלץ את הקבצים לתקייה כלשהי. לאחר מכן יש לפתוח cmd ולעבור לתקייה שאליה הקבצים חולצו דרך הפקודה cd.

לאחר מכן יש להריץ דרך פייתון את server.py

```
python server.py
```

למידע על פרמטרים ושימוש בהם ניתן להוסיף help-- להרצה.

גישת לקוחות

יש לפתוח דפדפן ולהקליד שם את פרטי השרת בשורת החיפוש בצורה- פורט:כתובת.
הערה: הדוגמאות הן בהתאם לשרת ברירת המחדל שמאזין על פורט 8080. אם זה שונה יש לשנותן בהתאם.
אם ניגשים מהמחשב שהשרת רץ עליו יש לכתוב: localhost:8080
אם ניגשים ממחשב ברשת המקומית של השרת ניתן לכתוב את כתובתו הווירטואלית:

(virtual IP):8080

ניתן לברר כתובת וויטואלית ע"י הרצת ipconfig/all ב-cmd במחשב השרת.
מכל מחשב ניתן לכתוב:

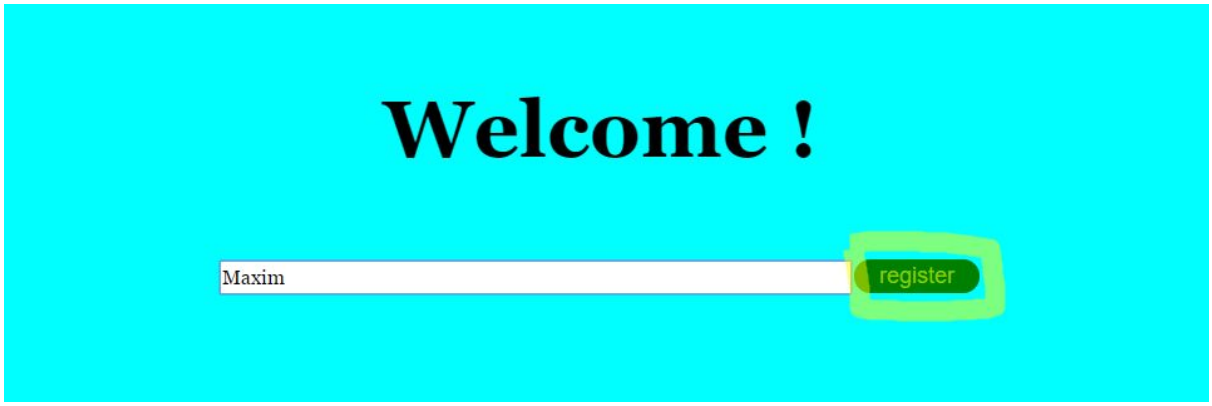
(real IP):8080

את הכתובת האמיתית ניתן לברר ע"י כניסה לאתר הבא ממחשב השרת:

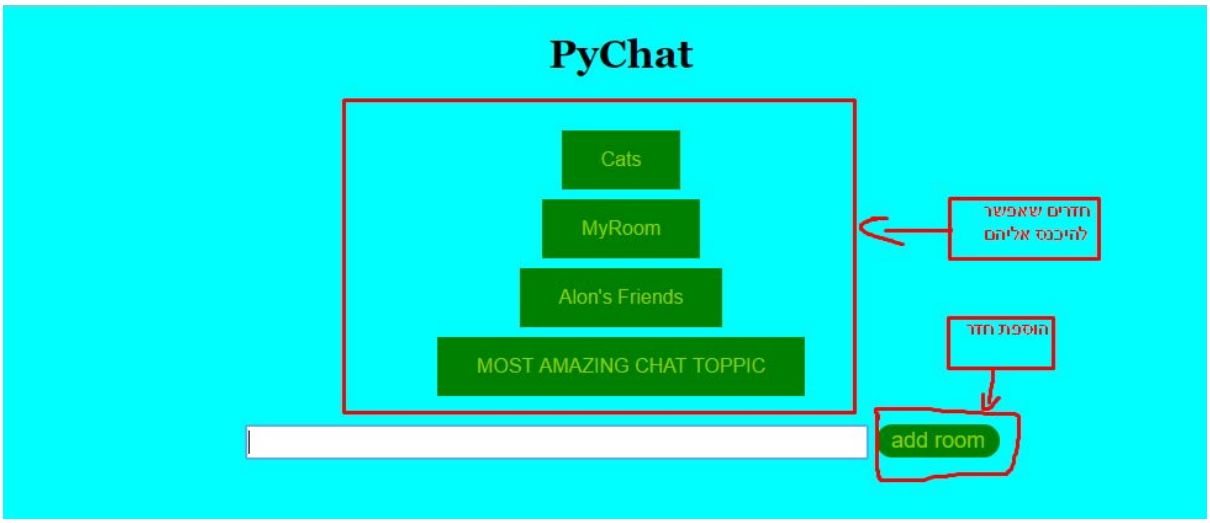
https://www.whatismyip.com/

הערה: בתרחיש אמיתי הנ"ל יפורסמו בצורה כזו או אחרת ללקוחות פוטנציאליים.

עם ההתחברות לשרת, בדף הראשי יש לבחור שם משתמש וללחוץ על register. תועברו לדף החדרים שם ניתן ליצור חדרים חדשים או להתחבר לחדרים קיימים ע"י לחיצה עליהם. יש לרשום הודעות לתוך חדר על מנת לתקשר עם משתמשים אחרים.
התחברות:



כניסה והוספת חדרים



שליחת הודעה בחדר צ'אט:

ליציאת מחדר צ'אט יש פשוט לסגור את הלשונית.

מגבלות ידועות

מעט מסורבל לגשת לשרת, במיוחד מרשת שאינה לוקאלית לרשתו. יש למצוא את כתובת ה-IP הווירטואלית והאמיתית של השרת וכן הפורט שעליו הוא מאזין בכדי שמשתמשים ידעו איך לגשת אליו.
כמובן שניתן לפתור זאת באמצעות השגת domain שעליו השרת יהיה מאוכסן.

חלק אישי

אין ספק שכתיבת הפרויקט דרשה מאמץ והשקעה רבה אך עם זאת הביאה עמה תחושת סיפוק ענקית ותובנות רבות לתוך עולם התכנות ותכנות ה-web.
קידוד הפרויקט דרש הבנה לעומק של הטכנולוגיות בהן הפרויקט ותכנון מקדים רב על מנת ליצור קוד פשוט ככל האפשר ולהקל על כתיבת הקוד עוד משלב מוקדם.
במסגרת כתיבת הפרויקט למדתי עצמאית שפות כמו javascrip ו-css וכן שיטות תכנות web כמו שימוש ב-XML בבקשות HTTP.
מימוש הפרויקט גרם לי להבין שכל בעיה מורכבת ניתן לפרק לחלקים פחות מורכבים וגם שלא ניתן לצפות הכל מראש ושצריך פשוט להתחיל לכתוב בשלב מסויים.
נהניתי לפתח את הלוגיקה שמאחורי הפרויקט ולראות את התוצר הגמור כמשהו מוחשי שניתן להשתמש בו.

תוכניות עתידיות

תכנית עתידית היא לממש את ערוץ התקשורת הסמוי שתואר בחלק הטכנולוגיות.

קוד פרויקט

https://github.com/maxim-mat/HTTP-Chat