תוכן עניינים:

מבוא:

בגלל שאני מאמין בתחבורה ציבורית, בגלל אהבתי הרבה לתחום התחבורה הציבורית, בגלל אהבתי העצומה לבאר שבע מולדתי ומכיוון שאני חבר ארגון "תחבורה בדרך שלנו" העוזרת לקדם תחבורה ציבורית במדינה, החלטתי במסגרת פרויקט הגמר בסייבר (תכנון ותכנות מערכות), לבנות ממשק הקשור לתחום התחבורה הציבורית.

לאחר לבטים רבים עם עצמי בחרתי בסופו של דבר בממשק תקשורת בין נהג בחברת האוטובוסים "דן באר שבע" (מפעילת התחבורה הציבורית בבאר שבע) לבין אחראי המשמרת באותה חברה (סדרן).

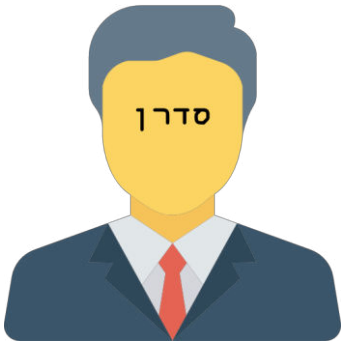
למערכת התקשורת שבניתי, ישנן מס' יכולות:

- כל הנהגים וכל הסדרנים הפועלים באותה משמרת יכולים לשוחח עם השני בחדר צ'אט ייעודי ולעדכן אחד את השני בכל מיני דברים הקשורים לתנועה ולאירועים מיוחדים בכבישים.
- סדרן יכול להורות לכל נהג איזה קו עליו לבצע והנהג יוכל לראות בצורה ויזואלית על מפה את מסלול הקו ולקבל את רשימת התחנות לידיו.

לצורך פיתוח GUI זה, בחרתי להשתמש בפלטפורמת Kivy ולהשתמש בשפת Python גרסה 3.7.

בחרתי בפרויקט הזה בכדי לאתגר את עצמי ובכדי לממש את אהבתי האידואולוגית לתחום התחבורה הציבורית לידי ביטוי ואני מאמין כי הצלחתי יפה מאוד.

הוראות למשתמש:



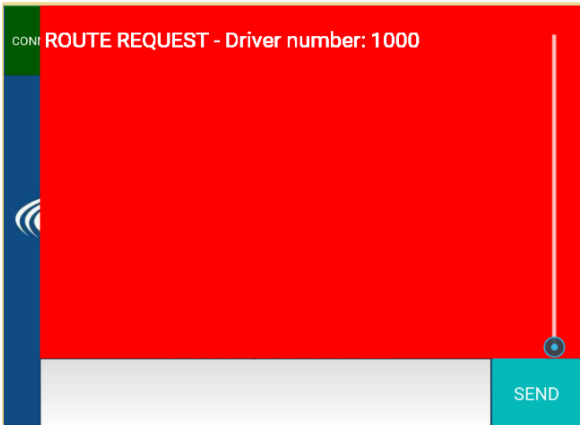
תחילה על הסדרן להתחבר לממשק וללחוץ על כפתור ה-CONNECT במסך.

לאחר מכן הנהג מתחבר למערכת, לוחץ על כפתור ה-CONNECT המופיע במסך, כאשר הוא לוחץ על הכפתור הנ״ל, השרת בוחר סדרן רנדמולי בקרב הסדרנים המחוברים לממשק.

אצל הסדרן תופיעה הודעה בחדר הצ׳אט האומרת כי מס׳ הנהג הנ״ל מחכה לפרטים של הקו שהוא צריך לבצע עכשיו.

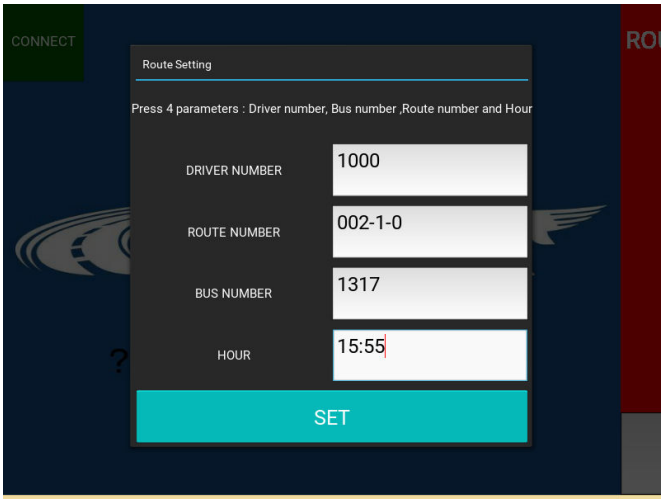
(מקרה קצה) – במידה ואין סדרן המחובר למערכת, מופיעה הודעה מותאמת אצל הנהג. כאשר מתחבר סדרן, על הנהג ללחוץ שוב על כפתור ה-CONNECT כדי לקבל קו.

השרת בודק האם מס׳ העובד שהוקש הוא מס׳ קיים במערכת ומוודא כי לא מודבר במספר פיקטיבי. בנוסף לכך אם גורם שלישי (פורץ) נכנס עם אותו מספר עובד למרות שהעובד כבר התחבר לממשק, אין אפשרות לאותו פורץ להתחבר – שני משתמשים עם אותו מס׳ עובד לא יכולים להיות מחוברים בו זמנית למשק.



כאשר הנהג מבקש לקבל קו, מופיעה ההודעה הנ"ל אצל הסדרנים המבקשת מן הסדרן לשגר קו לנהג הממתין בקצה השני.

לאחר מכן, על הנהג ללחוץ על המסך המרכזי של הממשק ויופיע לו ה-POPUP הבא:



לאחר מכן על הסדרן להקיש את הפרטים הבאים :

1. מס' הנהג שאליו הקו משויך.
2. מס' אוטובוס.
3. שעת יציאת הקו.
4. מספר הקו בשרשור הבא :

XXX-Y-Z

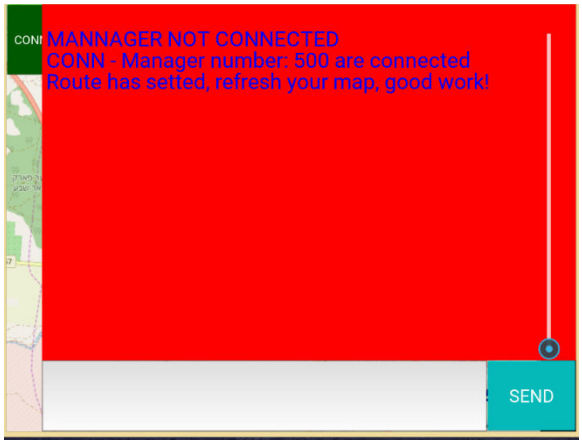
כאשר ה-X : מציין את מס' הקו, חובה להקיש מספר תלת ספרתי (לדוג' אם הקו הוא 2, יש להקיש 002).

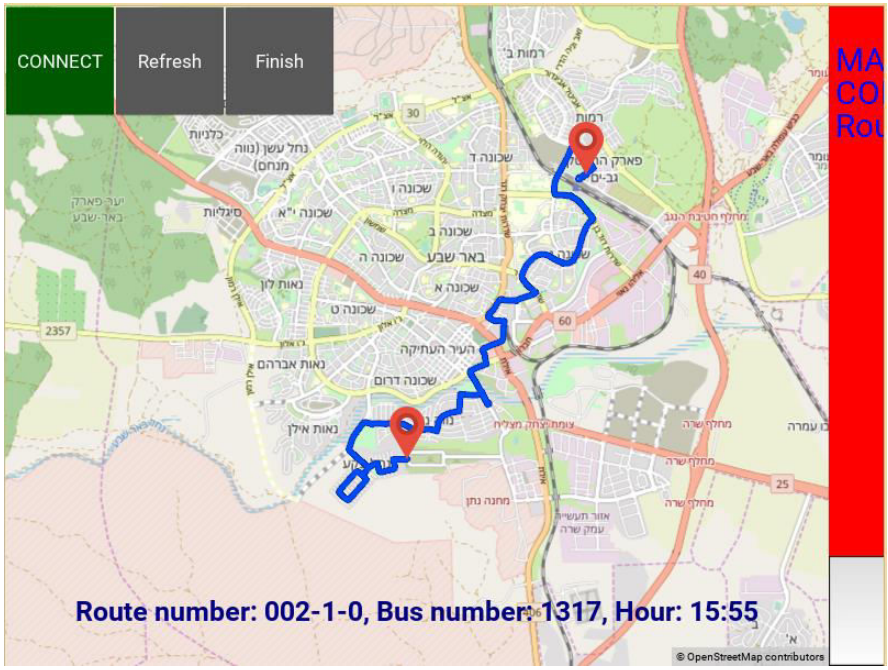
Y– כיוון הקו (יוצא מנקודת הנחה כי לסדרן יש מסמך המתאר את מס' הכיוון של כל קו).

Z – חלופת הקו (קווים רגילים ללא חלופה, החלופה תהיה שווה ל-0, אולם קווים שיש להם חלופה כגון קו 12א , החלופה תהיה בהתאם לכיוון הקו (1 או 2).

לדוגמה אם אני רוצה "לשגר" את קו 25 לנהג בכיוון למרכז ביג (כיוון 1) ובגלל שהקו הוא לא חלופה אזי מס' הקו יהיה : 025-1-0.

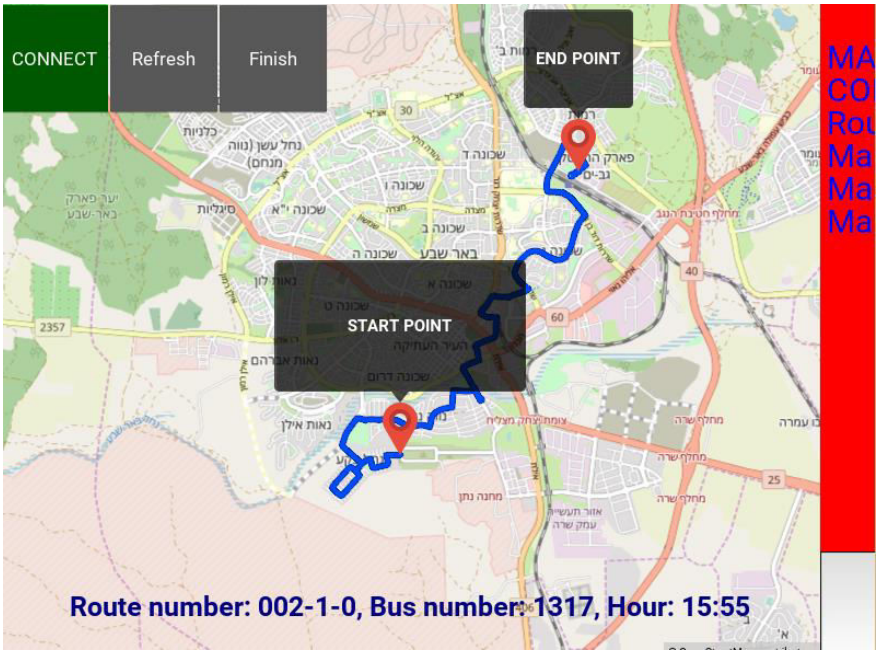
כאשר הנהג מקבל קו מהסדרן, תופיע לו הודעה מתאימה בחלון הצ׳אט, לאחר מכן עליו לעבור למסך השני (מסך המפה) ללחוץ על כפתור ה-REFRESH והקו יעודכן אצלו כפי שניתן לראות בתמונה הבאה.





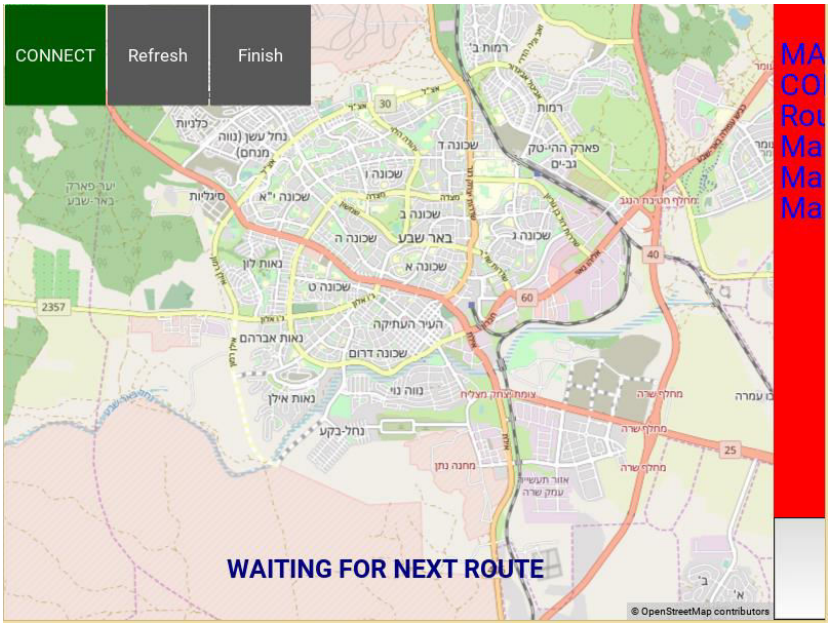
כפי שניתן לראות, למטה ישנה הודעה מתאימה האומרת לנהג איזה קו הוא מבצע, מס׳ האוטובוס עליו הוא נוהג ושעת יציאת הקו.

בנוסף לכך מופיעים על המפה שני סימונים על המפה שבלחיצה עליהם ניתן לראות מהי נקודת ההתחלה של הקו ומהי נקודת הסוף.

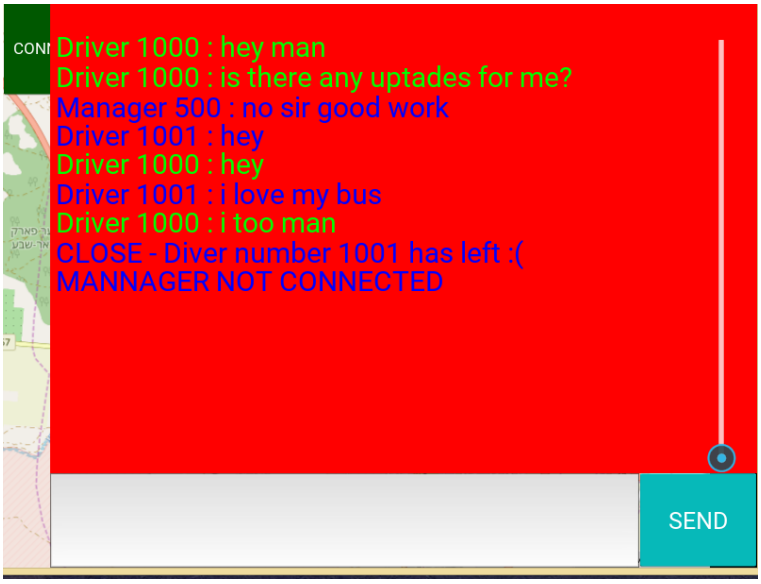


בסיום הקו על הנהג ללחוץ על כפתור ה- FINISH .

בעת לחיצת הכפתור, כל מה שקשור לקו הקודם על המפה נמחק והמערכת שולחת שוב הודעה לסדרן על כך שעליו לשגר קו חדש לנהג וחוזר חלילה.



בנוסף לכך, חדר הצ׳אט משמש כשמו כן הוא, משמש גם כחדר צ׳אט בין כל העובדים הנמצאים באותה משמרת.



קבצי GTFS:

אחד האתגרים הגדולים ביותר עבורי בהתחלה היה להבין בכלל מאיפה עליי לקחת את כל הנתונים אודות הקווים הקיימים בבאר שבע.

נוסף לכך עלו במוחי "שאלות פילוסופיות", מה זה קובץ של קו אוטובוס? האם זה קבצים בעלי פורמטים מיוחדים לעבודה עם מפות? האם יש בכלל קבצים או שמדובר בתוכנה שמציגה לך את כל הקווים בבת אחת? איך אני יכול להראות את הקבצים האלה על המפה?

לאחר חיפוש בגוגל, הבנתי כי כל הנתונים הקשורים אודות קווי האוטובוס במדינה נמצאים באגף המפתחים של אתר האינטרנט של משרד התחבורה.

ישנם שני סוגי קבצים, אחד הנקרא SIRI והוא אחראי בעצם על המידע בזמן אמת של קווי האוטובוס (מתי הקו מגיע לתחנה וכו' – **לא רלוונטי לעבודה**). השני נקרא GTFS והוא אחראי על המידע הסטטי של הקווים.

GTFS[1] – (ראשי תיבות General Transit Feed Specification) הינו הפורמט לצפייה בנתוני תחבורה ציבורית הכולל סט מידע על לוחות זמנים, מסלולים , תחנות ועוד **באמצעות קבצי טקסט** בפורמט בינ"ל אחיד. המידע בנוי בצורה שיש קשר בין הקבצים השונים וניתן לבצע ניתוחים שונים על סמך המידע המפורסם.

בעצם כך הבנתי כי כל קו מיוצג בעצם על ידי רשימה גדולה של קורדינטות (נ"צ) על המפה וחיבור בין כל נ"צ לנ"צ יוצר את מה שאנחנו רואים במפה בתור קו אוטובוס.

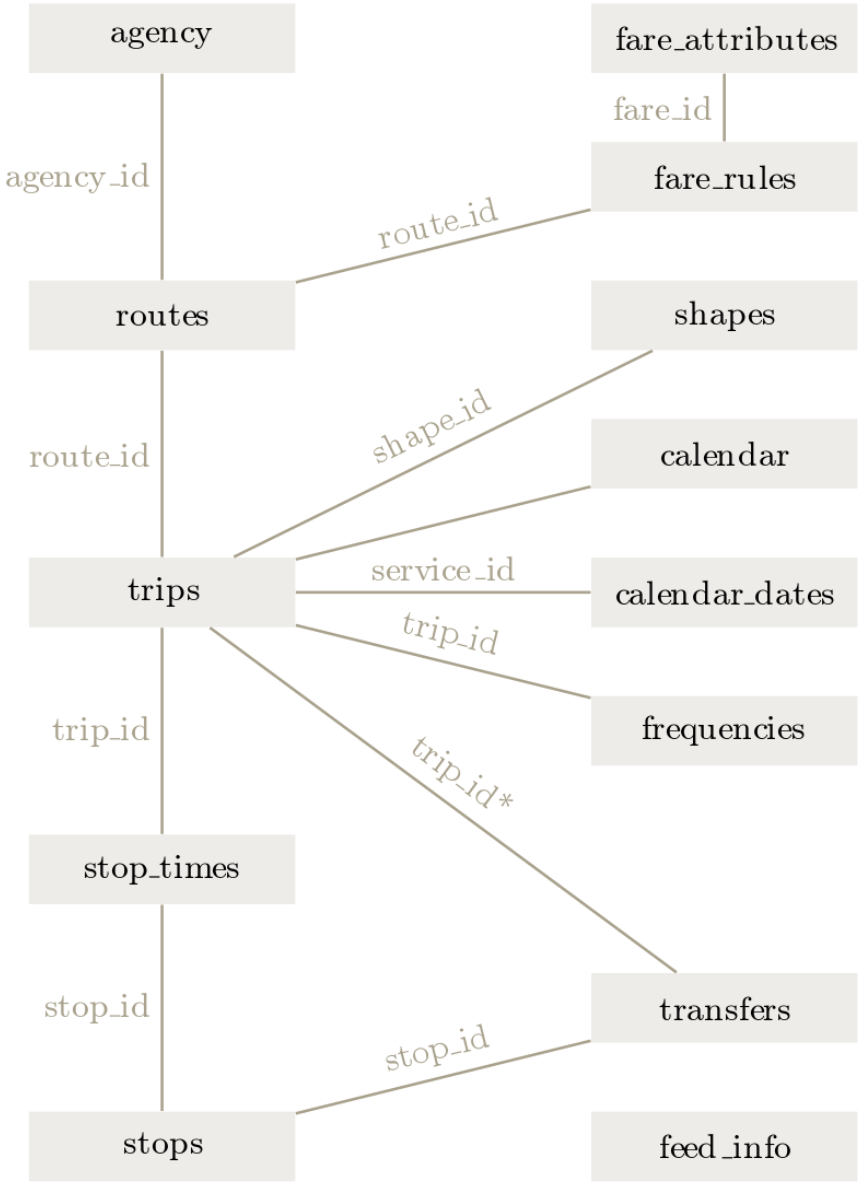
בתוך קבצי ה-GTFS ישנם קבצים רבים:

שם הקובץ	הסבר
agency	קובץ מפעילים
routes	קובץ מזהי הקווים (כל הכיוונים, וכל החלופות)
trips	קובץ נסיעות – רמת נסיעה בודדת
calendar	מציג את הימים בהם פועל הקו, ואת תאריכי הפעילות
times_stop	זמני נסיעות, לפי סדר תחנות הקו
stops	קובץ תחנות
shapes	קובץ מסלול הקו

[1] במידה ותרצו לקרוא עוד אודות קבצי GTFS של משרד התחבורה בישראל, תוכלו להיכנס לקישור הבא:
https://www.gov.il/BlobFolder/generalpage/gtfs_general_transit_feed_specifications/he/%D7%A1%D7%98%20%D7%A7%D7%91%D7%A6%D7%99%D7%9D%20-%20GTFS%20-%20%D7%94%D7%A1%D7%91%D7%A8%20%D7%9C%D7%9E%D7%A4%D7%AA%D7%97%D7%99%D7%9D_0.pdf

translations	קובץ תרגום
rules_Fare	קודי תעריף בין תחנות
attributes_Fare	מחיר נסיעה לקוד תעריף

בכל קובץ טקסט ישנם כמה איברים שקשורים בעצם לקבצים אחרים, את הקשר ניתן לתאר לפי הסרטוט הבא:



כך בעצם ניתן להבין על כל קו את כל הפרטים הרלוונטים לנו עליו (כמו שם המפעיל שלו, באיזה ימים הקו עובד, מה שעות היציאה של הקו וכו').

הקובץ החשוב ביותר בעבודה עבורי היה קובץ ה-SHAPES המכיל את כל נ"צ של מסלול הקו במפה (מסלול קו מוגדר כאוסף גדול של נ"צ שמותחים ביניהם קו במפה).

על מנת למיין לעצמי את הפרטים הקשורים לבאר שבע, ביצעתי באמצעות תוכנית שרשמתי בפייטון את הצעדים הבאים:

1. להיכנס לקבוץ הטקסט -AGENCY כדי לדעת מה הוא ה- AGENCY ID של חברת דן ב"ש – 32.
2. לדלות מתוך קובץ ה-ROUTS את כל הקווים והחלופות האפשריות של הקווים תחת המפעיל דן באר שבע.
3. לכל קו יצרתי תיקייה ששמה הוא מס' הקו בפורמט תלת ספרתי (לדוג' קו 2 יהיה בתיקייה ששמה 002, קו 25 יהיה בתיקייה ששמה 025 וכך הלאה).
4. לכל חלופה הייתי צריך לדלות את ה-SHAPE ID הרלוונטי שלה דרך קובץ ה-TRIPS .
5. עם ה-SHAPE ID שדליתי, הייתי צריך לאגד מקובץ ה-SHAPES את כל נ"צ הרלוונטיות לאותו SHAPE ID. רשמתי את כל הנ"צ של הקו במסמך טקסט, הכנסתי אותו לתיקייה המתאימה של הקו אליו הוא שייך ושיניתי את שם הטקסט לכיוון החלופה שהוא מייצג.
6. לכל נ"צ במסלול הקו יש מס' סידורי. מכיוון שבקובץ ה-SHAPES , נ"צ אינן מסודרת לפי מספרם הסידורי, היה צורך בכתיבת תוכנית הממיינת את נ"צ של הקו לאחר שהאחרון הוכנס כבר לקובץ הטקסט.

את כל זאת כאמור ביצעתי בשני קבצי "עיבוד" שרשמתי בפייטון **(קבצי העיבוד אינם חלק מהעבודה הרשמית אך הם מצורפים כאן כצילום מסך להעשרה בלבד)**

את שלבים 1-5 ביצעתי בקובץ הבא:

```python
import os

def createFolder(directory):
    try:
        if not os.path.exists(directory):
            os.makedirs(directory)
    except:
        pass


file_routes = open('routs_dan_bs.txt','r')
file_routs_string = file_routes.read()
file_routes.close()

file_shapes = open('shapes.txt','r')
file_shapes_string = file_shapes.read()
text_shapes_splited = file_shapes_string.split('\n')
file_shapes.close()

file_trips = open('trips.txt','r', encoding="utf8")
file_trips_string = file_trips.read()
text_trips_splited = file_trips_string.split('\n')
trips = []
for i in text_trips_splited:
    trips.append(i)
file_trips.close()
```

```python
text_lines_splited = file_routs_string.split('\n')

"""
routes file:

['17553', '32', '9', "hebrew_dest", '30009-1-0', '3', '']

index 0 = route id (Related to shapes.file and find the shape id of the route)
index 1 = const - agency number (dan beer sheva - 32)
index 2 = route number
index 3 = route destination (in hebrew - not relevant)
index 4 = code "###(route number) - (Direction of the route 1/2) - (alternative route - "halufa" - 1/2)
index 5 = const, route type (3 = bus)
index 6 = route color (regular route - " ", students route - "FF9933")
"""

R_T_S = []

for i in range(len(text_lines_splited)-1):
    route = text_lines_splited[i].split(',')
    route_id = route[0]
    r_t_s_in = [route_id]
    R_T_S.append(r_t_s_in)
for i in range(len(R_T_S)-1):
    f = False
    for trip_object in trips:
        if not f:
            trip_splited = trip_object.split(',')

            if R_T_S[i][0] == trip_splited[0]:
                R_T_S[i].append(trip_splited[5])
                f = True
        else:
            break

for i in range(len(R_T_S)-1):
    route = text_lines_splited[i].split(',')
    """route_number = route[2]
    try:
        if route[2][1] == 'א':
            route_number = route[2][0]
        elif route[2][2] == 'א':
            route_number = route[2][:2]
    except:
        pass"""
    R_T_S[i].append(route[4][-7:])

    #createFolder('./'+route_number+'/')

for c in range(len(R_T_S)-1):
    path = "C:\\Users\\ariel\\Desktop\\Pro\\GTFS\\route\\"
    new = ""
    for i in range(len(text_shapes_splited)):
        y = text_shapes_splited[i].split(',')
        try:
            if y[0] == R_T_S[c][1]:
                new = new + text_shapes_splited[i] + '\n'
        except:
            pass

    path_name = R_T_S[c][2][:3] + "\\" + R_T_S[c][2][-3:]+'.txt'
    path += path_name
    with open(path,'w') as f:
        f.write(new)


print(R_T_S)
print("stop")
```

את השלב ה-6 ביצעתי באמצעות הקובץ הבא:

```python
import os
import glob

path = 'C:\\Users\\ariel\\Desktop\\Pro\\GTFS\\route\\'

def takeLast(elem):
    return elem[-1]

for filename in os.listdir(path):
    print(filename)
    path += filename
    for textfiles in os.listdir(path):
        file = open(path+'\\'+textfiles,'r')
        file_string = file.read()
        splited = file_string.split('\n')
        new = []
        for i in splited:
            if i != '':
                splitex_2 = i.split(',')
                num_string = splitex_2[-1]
                num_int = int(num_string)
                splitex_2[-1] = num_int
                new.append(splitex_2)
        new.sort(key = takeLast)
        fill = ''
        for i in new:
            joined_text = ",".join(str(x) for x in i)
            fill += (joined_text +'\n')

        file2 = open(path+'\\'+textfiles,'w')

        file2.write(fill)

    path = 'C:\\Users\\ariel\\Desktop\\Pro\\GTFS\\route\\'
```

כאמור, קבצים אלה אינם כחלק מהעבודה הרשמית שכן מטרתי בבניית ממשק זה מלכתחילה בעיר הייתה בניית ממשק עם מידע סטטי כל הקווים הנמצאים **כיום** בבאר שבע אבל עם קצת שינויים בקוד, ניתן לדלות מקבצי GTFS המדיניים את הקווים של כל מפעיל תחבורה ציבורית שרק נרצה.

הוראות למתכנת:

שם	תאריך שינוי	סוג	גודל
cache	27/05/2019 21:35	תיקיית קבצים	
server	27/05/2019 21:33	תיקיית קבצים	
driver	27/05/2019 20:47	JetBrains PyChar...	16 KB
manager	27/05/2019 18:13	JetBrains PyChar...	10 KB
my_background	23/04/2019 13:42	תמונה מסוג JPEG	32 KB

לממשק יש שני סוגי לקוחות – נהג, סדרן ושרת אחד.

בתוך תיקיית ה-SERVER מצויים ארבעה קבצים:

1. קובץ הפייטון של השרת.
2. קובץ טקסט של מס' העובדים של הנהגים (DRIVERS)
3. קובץ טקסט של מס' העובדים של הסדרנים (MANAGERS)
4. תיקיית ROUTE

חשוב להדגיש כי המערכת שלי היא הינה מערכת לניהול נהגים, אין אופציה להוסיף\להסיר נהג מהמערכת. הממשק יוצא מהנחה שישנה תוכנת ניהול שמייצאת במסמך טקסט את מספרי העובדים וכבר עשתה להם HASH, הממשק שלי יוצא מהנחה שקבצי הטקסט עם מספרי העובדים כבר אינם מוצפנים.

BUS_LAN ◂ server ◂ חפש ב- server

שם	תאריך שינוי	סוג	גודל
route	27/05/2019 20:46	תיקיית קבצים	
drivers	13/05/2019 20:25	מסמך טקסט	1 KB
managers	13/05/2019 20:25	מסמך טקסט	1 KB
server1	27/05/2019 21:47	JetBrains PyChar...	7 KB

בקבצי הטקסטים מצויים מס' העובדים שהמנהלים יכולים לעדכן כל פעם (השרת לוקח כל פעם את הנתונים אודות מס' העובדים מן הקבצים הללו).

בתיקיית ה-ROUTE מצויים כל תיקיות הקווים של המפעילה דן ב"ש בבאר שבע, כאמור בפרק הקודם, כל תיקייה מציינת מס' קו ובתוכה מצואיים קבצי טקסטים המציינים את החלופות של הקו (כל התיקיות והקבצים האלה יוצרו על ידי התכנית שציינתי בפרק הקודם).

תחילה יש להפעיל את השרת ורק אז להפעיל את ממשק הנהג / סדרן.

השרת:

כללי השרת – השרת מקבל מידע מהלקוחות. כל מידע שהוא מקבל מהשרת הוא ממיין לפי שרשור האותיות המופיעות בתחילת ה-DATA שקיבל.

כעת נעבור על הפונקציה החשובה ביותר בשרת והיא הפונקציה שמטפלת בכל המידע שמגיע לשרת (פונקציית handle_client).

```
# This function always works in the background and handles all information that comes from customers
# and runs other functions according to information received from the customer.
def handle_client(conn):
    while True:
        try:
            data = conn.recv(1024)
            data_decoded = data.decode()
            data_splited = data_decoded.split(' ')

            if data_decoded[:6] == 'Number':
                first_connect_from_manager(data_splited)
            elif data_decoded[:5] == 'Route':
                route_send(data_splited, data_decoded)
            elif data_decoded[:5] == 'ROUTE':
                worker_number = data_splited[-1]
                send_to_random_manager(data, worker_number)
            elif data_decoded[:5] == 'CLOSE':
                worker_number = data_splited[4]
                send_to_all(data, worker_number)
                del all_addresses_wmnumber[worker_number]
            else:
                worker_number = data_splited[1]
                send_to_all(data, worker_number)
        except:
            pass
```

כעת נעבור על סוגי המידע השונים שהשרת יכול לקבל:

1. שרשור המתחיל ברצף Number נשלח לשרת בעת שלקוח רוצה להתחבר, הפונקצייה מפעילה את פונצקיית first_connect_from_manager.
2. שרשור המתחיל ברצף Route נשלח לשרת תמיד מהסדרן שרוצה לשלוח קו לנהג, הפונקצייה מפעילה את פונקציית route_send.
3. שרשור המתחיל ברצף ROUTE נשלח לשרת תמיד מנהג המבקש שסדרן כלשהו ישלח לו קו, הפוקנצייה מפעילה את פונקציית send_to_random_manager.
4. שרשור המתחיל ברצף CLOSE נשלח לשרת בעת שלקוח רוצה להתנתק, הפונקצייה מוחקת את הלקוח ממילון המשתמשים all_addresses_wmnumber ושולחת הודעה מתאימה למשתמשים המחוברים.
5. כל מידע אחר שמגיע, הפונקצייה מפעילה את פונקציית send_to_all השולחת את המידע הנ"ל לכל המשתמשים המחוברים מלבד המשתמש ששלח את המידע.

כעת נעבור על הפונקציות שציינתי לעיל.

```
# When a new client connects to the system, this function checks whether the driver is already connected or not.
# In addition, the function adds the client who joined the workers dictionary - all_addresses_wmnumber.
def first_connect_from_manager(data_splited):
    number = data_splited[2]
    worker_type = data_splited[1]
    if worker_type == 'M':
        if exist(number):
            conn.send('False'.encode('utf-8'))
        else:
            b = False
            for i in manager_Numbers:
                if i == number and not b:
                    b = True
                    all_addresses_wmnumber[number] = [conn, addr[0]]
                    all_managers.append(number)
                    conn.sendall('True'.encode('utf-8'))
                    break
            if not b:
                conn.send('False'.encode('utf-8'))
    else:
        if exist(number):
            conn.send('False'.encode('utf-8'))
        else:
            b = False
            for i in driver_Numbers:
                if i == number and not b:
                    b = True
                    all_addresses_wmnumber[number] = [conn, addr[0]]
                    conn.sendall('True'.encode('utf-8'))
                    break
            if not b:
                conn.send('False'.encode('utf-8'))


# func for first_connect_from_manager func
# The func receives the workers number
def exist(n):
    numbers = all_addresses_wmnumber.keys()
    for i in numbers:
        if n == i:
            return True
    return False
```

פונקציית first_connect_from_manager מקבלת כפרטמר את המידע שנשלח לשרת כשהוא כבר עבר פונקציית SPLIT . הפונקצייה תחזיר ללקוח False האם מס׳ הנהג כבר מחובר למערכת / מספרו לא תקני. הפונקצייה תשלח True אם מס׳ הנהג תקני והוא לא מחובר כבר למערכת.

הפונקצייה משתמשת בפונקציית העזר exist הבודקת אם עובד שמספרו N מחובר.

```python
# data_decoded will be in this form: "Route _drivernumber_ _routenumber_ _busnumber_ _hour_"
# data_splited will be data_decoded.split(' ')
# The function sends all the coordinates of the line that the manager requested send to the driver.
def route_send(data_splited, data_decoded):
    try:
        driver = all_addresses_wmnumber[data_splited[1]]
        conn_driver = driver[0]
        route_number_splited = data_splited[2].split('-')
        path = 'route\\' + route_number_splited[0] + "\\" + route_number_splited[1] + "-" + route_number_splited[
            2] + '.txt'
        with open(path, 'r') as coordinates_file:
            to_send = data_decoded + "\n"
            coordinates_file_string = coordinates_file.read()
            coordinates_file.close()
            to_send += coordinates_file_string
        conn_driver.sendall(to_send.encode('utf-8'))
    except Exception as e:
        print(e)
```

פונקציית route_send מקבלת כפרמטר את המידע שהתקבל מהסדרן, מהמידע הזה הפונקצייה מאתרת את קובץ הקו הרצוי בתיקיית ה-route ושולחת את תוכן הקובץ (אוסף של קורדינטות) לנהג.

```python
# When a driver requests a line,
# this function sends the request to a random scheduler that is connected to the system.
# The func receives the driver's number
def send_to_random_manager(data, worker_number):
    conn_worker = all_addresses_wmnumber[worker_number][0]
    if is_manager_connected():
        conn_worker.send('MANNAGER IS CONNECTED'.encode('utf-8'))
        random_mannager = random.choice(all_managers)
        while not exist(random_mannager):
            random_mannager = random.choice(all_managers)
        random_mannager_contact = all_addresses_wmnumber[random_mannager]
        random_mannager_contact[0].sendall(data)
    else:
        conn_worker.send('MANNAGER NOT CONNECTED'.encode('utf-8'))
```

פונקציית send_to_random_manager שולחת בקשה לסדרן רנדמולי לשגר קו לנהג שמחכה כעת לקו.

הלקוח – נהג \ סדרן:

דרישות מערכת: יש לוודא כי החבילות הבאות מותקנות על המחשב:

1. kivy בגרסה 1.10.1
2. Kivy-garden בגרסה 0.1.4
3. להתקין את התוסף mapview[2] דרך kivy-garden.
4. Pillow בגרסה 6.0.0

כאמור, ישנם שני צדדי לקוח – סדרן ונהג. שני ה-GUI הנ"ל רשומים בשפת KIVY ולכן בתחילת שני התכניות הנ"ל מופיעים קטעים בשפת KV האחראים לעיצוב ה-GUI.

```
#:import MapSource kivy.garden.mapview.MapSource

<Login>:
    BoxLayout
        id: login
        orientation: 'vertical'

        spacing: 20

        Label:
            text: 'Welcome!'
            font_size: 32
        Label:
            text: 'Please enter your driver number'
            font_size: 24

        BoxLayout:
            orientation: 'vertical'

            Label:
                text: 'Driver Number'
                font_size: 18
                halign: 'left'
                text_size: root.width-20, 20

            TextInput:
                id: driverNumber
                multiline:False
                font_size: 28

        Button:
            text: 'Connection'
```

כעת נעבור על מס' פוקנציות/מחלקות משמעותיות משותפות לשני הלקוחות.

[2] ניתן להתקין את התוסף דרך PIP דרך הקישור הבא: https://github.com/kivy-garden/garden.mapview

```python
# this class is responsible for the LOGIN screen and for proper connection with the server
class Login(Screen):

    # the function connect to server
    def do_login(self, driverNumber):
        self.resetForm()

        global client
        client = socket.socket()
        client.connect((SERVERIP, PORTNUM))

        global dNumber
        dNumber = driverNumber

        self.manager.transition = SlideTransition(direction="left")

        if self.check_work_number():
            self.manager.current = 'connected'

        else:
            dNumber = 0
            self.manager.current = 'retry'

    def resetForm(self):
        self.ids['driverNumber'].text = ""

    def getDriverNumber(self):
        global dNumber
        return dNumber

    # the funftion return true if the driver number is correct and return false if the driver number is incorrect
    # (if the driver is allready connected, the func will also return false)
    def check_work_number(self):
        data_send = ("Number D " + dNumber).encode()
        client.sendall(data_send)
        client_input = client.recv(1024)
        decoded_input = client_input.decode()
        if decoded_input == 'True':
            return True
        else:
            return False
```

המחלקה LOGIN אחראית להתחבר לשרת ולוודא כי מס' הנהג שהוקש הוא אכן מס' נכון וקיים – במידה ולא, מופיעה הודעה מותאמת על המסך. (הפונקצייה שמבצעת את הבדיקה עם השרת הינה פונקציית check_work_number.

המחלקה העיקרית בשני הלקוחות היא מחלקת Connected.

מכיוון שכעת יהיו הבדלים בין שני סוגי הלקוחות, אעבור כעת על כל מחלקה כזו בנפרד אצל כל סוג לקוח.

נתחיל אצל המחלקה Connected אשר נמצאת אצל הנהג (driver.py).

```python
class Connected(Screen):
    def __init__(self, **kwargs):
        super(Connected, self).__init__(**kwargs)
        self.k = False

    #the func main purpose is to start threading and request a route from random manager that online now- worked when the worker click "CONNECT" button
    def start_threading(self):
        Connected.scrolling(self)
        client.sendall(("ROUTE REQUEST - Driver number: " + Login.getDriverNumber(self)).encode('utf-8'))
        if not self.k:
            t = Thread(target=Connected.get_message_thread, args=(self, client))
            t.daemon = True
            t.start()
        self.k = True

    #when the driver finish his route, the func delete all layers from the previous line and request new route- worked when the worker click "Finish" button
    def finish_route(self):
        try:
            self.ids.mapscreen.remove_layer(MAP_LAYERS[0])
            del MAP_LAYERS[:]
        except:
            pass
        self.ids.start_marker.lon = 20.251738
        self.ids.start_marker.lat = 20.251738
        self.ids.end_marker.lon = 20.251738
        self.ids.end_marker.lat = 20.251738
        client.sendall(("ROUTE REQUEST - Driver number: " + Login.getDriverNumber(self)).encode('utf-8'))
        route_shirshur = '[b]' + '[color=000080]WAITING FOR NEXT ROUTE[/color]' + '[/b]'
        self.ids['route_label'].text = route_shirshur
        self.refresh_map()
```

פונקציית start_threading מפעילה את ה-THREAD שאחראי להעפיל ברקע תמיד את הפונקצייה הדואגת לקבל הודעות מהשרת (get_message_thread) והפונקצייה דואגת לשלוח לשרת בקשה לכך שסדרן ישגר קו לאותו נהג.

פונקציית finish_route גורמת להעלמת הקו הישן מן המפה (כולל הפופאפים) ומבקשת שוב שיגור קו מהשרת.

```python
#message send - worked when the worker click "SEND" button
def send(self, text_input):
    self.ids['text_input'].text = ""

    to_send = ("Driver " + Login.getDriverNumber(self) + " : " + text_input).encode()

    client.sendall(to_send)

    massage = ChatText(text=to_send, size_hint_y=None, height=52, font_size=30, background_color=(0, 0, 0, 0),
                       cursor_color=(1, 1, 1, 1), foreground_color=(0, 1, 0, 1))
    massage.text_size = (massage.size)
    self.ids.layout2.add_widget(massage)
```

פונקציית send דואגת לשליחת ההודעה בחדר הצ׳אט שהנהג הקיש לכלל העובדים המחוברים.

```python
# this func is always works(because of the thread). this func receives messages from the server
# if the server send data the strats with "Route" so the function causes the line to appear on the map
# if the server send data the are not strats with "Route" so this func add the message to the chatbox
def get_message_thread(self, client):
    while True:
        client_input = client.recv(9000000)
        decoded_input = client_input.decode("utf-8")

        if decoded_input[:5] == 'Route':
            line = LineMapLayer(decoded_input)
            start_coor, end_coor = line.first_last_coordinate()

            MAP_LAYERS.append(line)
            splited_input = decoded_input.split(" ")
            route_shirshur = '[b]' + '[color=000080]Route number: ' + splited_input[2] + ", Bus number: " + \
                             splited_input[3] + ", Hour: " + splited_input[4][0:5] + '[/color]' + '[/b]'
            self.ids['route_label'].text = route_shirshur
            massage_text = "Route has setted, refresh your map, good work!"
            massage = ChatText(text=massage_text, size_hint_y=None, height=50, font_size=30,
                               background_color=(0, 0, 0, 0), cursor_color=(255, 255, 255, 255),
                               foreground_color=(0, 0, 1, 1))
            massage.text_size = (massage.size)
            self.ids.layout2.add_widget(massage)
            self.ids.mapscreen.add_layer(line, mode="scatter")

            self.ids.start_marker.lon = start_coor[1]
            self.ids.start_marker.lat = start_coor[0]
            self.ids.end_marker.lon = end_coor[1]
            self.ids.end_marker.lat = end_coor[0]
        else:
            if decoded_input =='MANNAGER IS CONNECTED':
                route_shirshur = '[b]' + '[color=000080]WAITING FOR NEXT ROUTE [/color]' + '[/b]'
                self.ids['route_label'].text = route_shirshur
            else:
                massage = ChatText(text=decoded_input, size_hint_y=None, height=50, font_size=30,
                                   background_color=(0, 0, 0, 0), cursor_color=(255, 255, 255, 255),
                                   foreground_color=(0, 0, 1, 1))
                massage.text_size = (massage.size)
                self.ids.layout2.add_widget(massage)
```

הפונקצייה המרכזית הינה פונקציית get_message_thread הפועלת ברקע. לפונקצייה יש שני מצבים, אם המידע שהיא מקבלת מתחיל בשרור ROUTE הרי מדובר בהודעה בה מצויינים כל נ״צ של הקו על מנת להציג אותם על מפת הנהג, הפונקצייה דואגת ״למקם״ את הקו על המפה, לרשום מהי נקודת הפתיחה/סיום של הקו ולשלוח הודעה מתאימה לנהג.

כל הודעה אחרת שהנהג מקבל, יש שוב שני אופציות.

במידה והשרשור אומר שאין מנהלים מחוברים (מקרה קצה) אז המערכת תדאג להודיע על כך לנהג.

כל הודעה אחרת, הרי מדובר בהודעת צ׳אט רגילה אשר תופיע בחדר הצ׳אט של הנהג.

```python
# this class is causes the line to appear on the map
class LineMapLayer(MapLayer):
    def __init__(self, coordinates_file_string, **kwargs):
        super(LineMapLayer, self).__init__(**kwargs)
        self._coordinates = []
        self.zoom = 0

        splited = coordinates_file_string.split('\n')
        coordinates = []

        for i in splited:
            splited_text = i.split(',')
            try:
                cooridnate1 = []
                cooridnate1.append(float(splited_text[1]))
                cooridnate1.append(float(splited_text[2]))
                coordinates.append(cooridnate1)
            except:
                pass

        self.coordinates = coordinates

    @property
    def coordinates(self):
        return self._coordinates

    @coordinates.setter
    def coordinates(self, coordinates):
        self._coordinates = coordinates

        #: Since lat is not a linear transform we must compute manually
        self.line_points = [(self.get_x(lon), self.get_y(lat)) for lat, lon in coordinates]

    # self.line_points = [mapview.get_window_xy_from(lat, lon, mapview.zoom) for lat, lon in coordinates]

    def first_last_coordinate(self):
        length = len(self._coordinates)
        return self._coordinates[0], self._coordinates[length-1]
```

מחלקת LineMapLayer אחראית על כך שכל נ״צ שנשלחו למחלקה יופיעו בצורה מסודרת בתור קו במפה. המחלקה מקבלת כפרטמר את הקורדינטות של הקו בתור STRING.

המחלקה יורשת ממחלקת MapLayer אשר נמצאת בחבילה של kivy garden mapview.

בקובץ הסדרן (manager.py) מבנה הקוד דומה לקובץ הנהג (driver.py) מלבד מסך המפה שקיים אצל הנהג ואינו קיים אצל הסדרן לכן אין צורך לחזור על ההסברים בשנית.