In [3]:
```
import warnings
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.dates as mdate
import seaborn as sns
from pandas import datetime
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
```

In [4]:
```
color = sns.color_palette()
print('Please wait. Importing data...')
df = pd.read_excel('C:/Users/ThinkPad/Desktop/Masterarbeit/PM10_2019.xlsx',header=0,encoding='utf
print('import completed.')

#def date_parser(x):
#    return datetime.strptime(x, "%d.%m.%Y %H:%M")
df[['date','time']] = df['Zeitpunkt'].str.split(expand=True)
df['datetime'] = (pd.to_datetime(df.pop('date'), format='%d.%m.%Y') +
                  pd.to_timedelta(df.pop('time') + ':00'))
print (df)
```

```
Please wait. Importing data...
import completed.
             Zeitpunkt Andechs/Rothenfeld Ansbach/Residenzstraße  \
0     01.01.2019 01:00                  5                     88
1     01.01.2019 02:00                  8                     35
2     01.01.2019 03:00                 10                     25
3     01.01.2019 04:00                 13                     28
4     01.01.2019 05:00                 13                     24
...                ...                ...                    ...
8755  31.12.2019 20:00                 15                     41
8756  31.12.2019 21:00                 18                     32
8757  31.12.2019 22:00                 19                     56
8758  31.12.2019 23:00                 19                     80
8759  31.12.2019 24:00                 18                     81

     Augsburg/Bourges-Platz Augsburg/Karlstraße Augsburg/Königsplatz  \
0                       300                 483                  322
1                        99                 103                   93
2                        56                  65                   50
3                        45                  44                   41
4                        33                  31                   19
...                     ...                 ...                  ...
8755                    130                  76                   77
8756                     69                  74                   71
8757                     60                  50                   41
8758                     58                  57                   46
8759                     92                  61                   59

     Augsburg/LfU Bad Hindelang/Oberjoch Bamberg/Löwenbrücke  \
0             122                     7                 154
1             122                     9                  41
2             139                     7                  34
3              38                     2                  40
4              22                     2                  45
...           ...                   ...                 ...
8755          112                     2                  60
8756          106                     2                  71
8757           74                     2                  78
8758           51                     2                  78
8759           42                     2                  97

     Bayreuth/Hohenzollernring  ... Passau/Stelzhamerstraße  \
0                          614  ...                     24
1                          144  ...                     58
2                           30  ...                     59
3                           24  ...                     66
4                           23  ...                     51
...                        ...  ...                    ...
8755                        43  ...                     55
8756                        78  ...                     53
8757                        84  ...                     54
```

```
8758                      81  ...                      60
8759                     153  ...                      68

     Regensburg/Rathaus Schwabach/Angerstraße Schweinfurt/Obertor  \
0                   241                   277                 991
1                    83                    39                 511
2                    29                    34                  31
3                    25                    39                  29
4                    33                    17                  31
...                 ...                   ...                 ...
8755                 62                    48                  65
8756                 66                    52                  57
8757                 73                    70                  45
8758                 54                    90                  44
8759                 86                    82                  42

     Sulzbach-Rosenberg/Lohe Tiefenbach/Altenschneeberg  \
0                         44                          3
1                         33                          3
2                         39                          4
3                         37                          4
4                         53                          5
...                      ...                        ...
8755                      38                         10
8756                      45                         17
8757                      37                         18
8758                     110                          9
8759                     108                          7

     Trostberg/Schwimmbadstraße Würzburg/Kopfklinik Würzburg/Stadtring Süd  \
0                           111                452                    125
1                            39                 80                     73
2                            28                 56                     32
3                            28                 55                     30
4                            32                 49                     26
...                         ...                ...                    ...
8755                         29                 49                     58
8756                         41                 39                     42
8757                         41                 45                     61
8758                         35                 49                     77
8759                         33                 52                    176

                datetime
0    2019-01-01 01:00:00
1    2019-01-01 02:00:00
2    2019-01-01 03:00:00
3    2019-01-01 04:00:00
4    2019-01-01 05:00:00
...                  ...
8755 2019-12-31 20:00:00
8756 2019-12-31 21:00:00
8757 2019-12-31 22:00:00
8758 2019-12-31 23:00:00
8759 2020-01-01 00:00:00

[8760 rows x 34 columns]
```

In [18]:
```
data = df[['datetime', 'Augsburg/Königsplatz']]
#data['date'] = data['Zeitpunkt'].map(lambda x: date_parser(x))
data['pm10']=data['Augsburg/Königsplatz']
print(data)
data.pm10.describe()
```

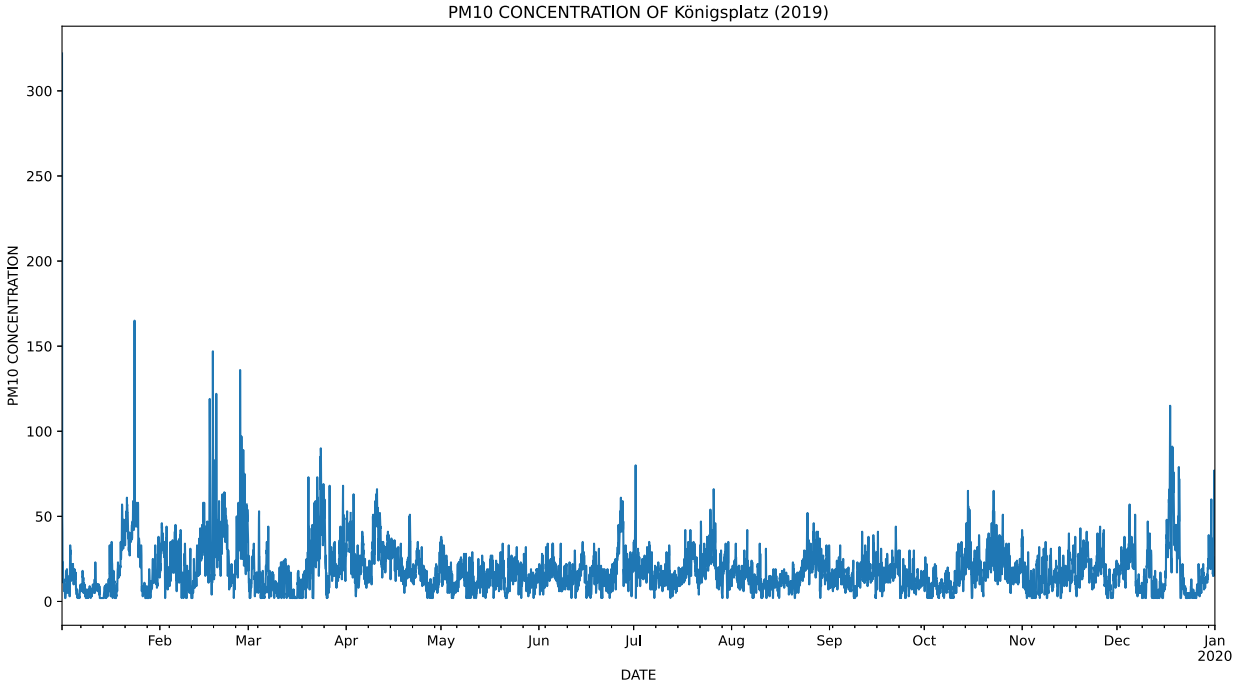
```
                datetime Augsburg/Königsplatz pm10
0    2019-01-01 01:00:00                  322  322
1    2019-01-01 02:00:00                   93   93
2    2019-01-01 03:00:00                   50   50
3    2019-01-01 04:00:00                   41   41
4    2019-01-01 05:00:00                   19   19
...                  ...                  ...  ...
8755 2019-12-31 20:00:00                   77   77
8756 2019-12-31 21:00:00                   71   71
8757 2019-12-31 22:00:00                   41   41
8758 2019-12-31 23:00:00                   46   46
8759 2020-01-01 00:00:00                   59   59

[8760 rows x 3 columns]
```

Out[18]:
```
count     8760
unique     100
top          2
freq       443
Name: pm10, dtype: int64
```

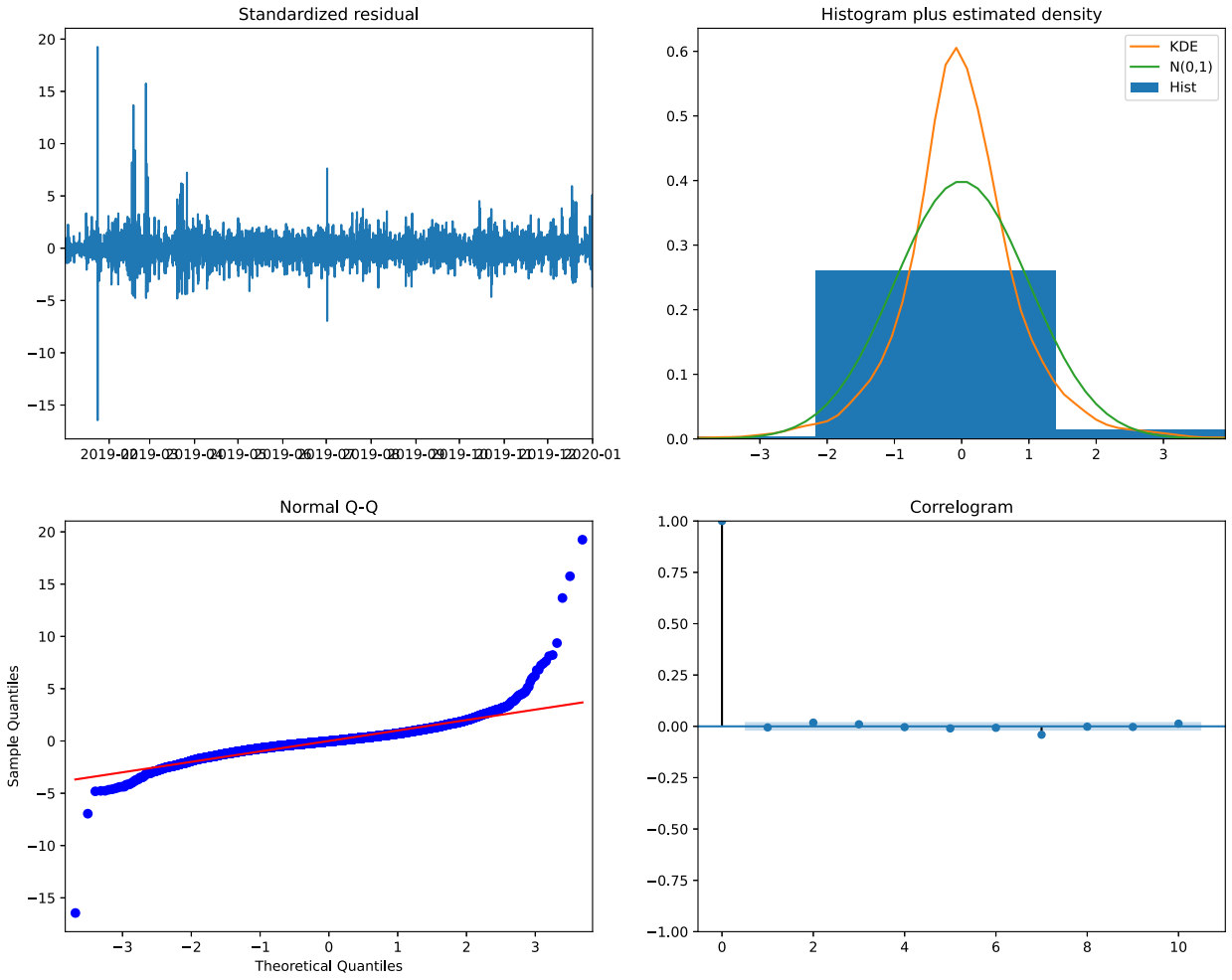
In [20]:
```
data=data[['datetime','pm10']]
data['pm10'] = data['pm10'].map(lambda x: str(x))
data['pm10'] = pd.to_numeric(data['pm10'],errors='coerce')
data.index = data['datetime']
#data['datetime'] = pd.to_datetime(data['datetime'])
#data.set_index('datetime')
#data['data.apply(pd.to_numeric)'] = pd.to_numeric(data['pm10'])

data.pm10.plot(figsize=[15, 8])
plt.xlabel("DATE")
plt.ylabel("PM10 CONCENTRATION")
plt.title("PM10 CONCENTRATION OF Königsplatz (2019)")
plt.show()
```



In [21]:
```
mod = sm.tsa.statespace.SARIMAX(data.pm10,
order=(1, 1, 1),
seasonal_order=(1, 0, 1, 12),
enforce_stationarity=False,
enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
results.plot_diagnostics(figsize=(15, 12))
plt.show()
```

```
D:\ANACONDA\lib\site-packages\statsmodels\tsa\base\tsa_model.py:159: ValueWarning: No frequency in
formation was provided, so inferred frequency H will be used.
  warnings.warn('No frequency information was'
D:\ANACONDA\lib\site-packages\statsmodels\tsa\base\tsa_model.py:159: ValueWarning: No frequency in
formation was provided, so inferred frequency H will be used.
  warnings.warn('No frequency information was'
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7589      0.005    152.873      0.000       0.749       0.769
ma.L1         -0.9479      0.004   -257.387      0.000      -0.955      -0.941
ar.S.L12       0.0699      0.057      1.222      0.222      -0.042       0.182
ma.S.L12      -0.0423      0.057     -0.740      0.459      -0.154       0.070
sigma2        35.0564      0.120    291.760      0.000      34.821      35.292
==============================================================================
```

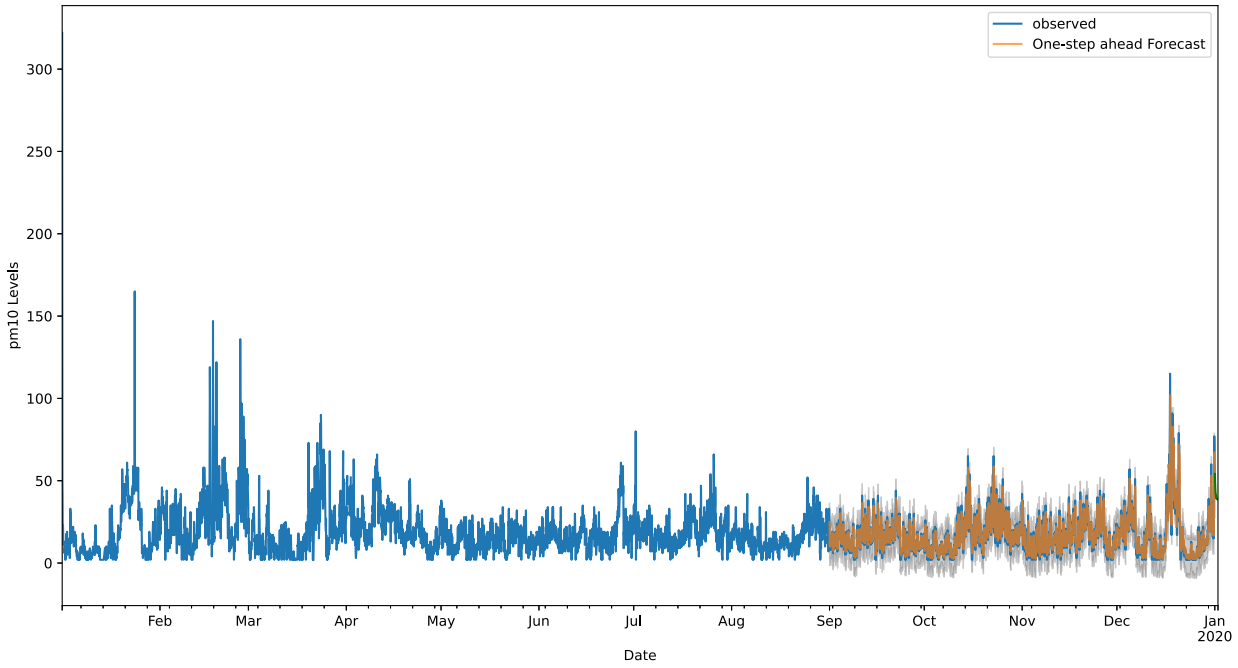


In [26]:
```
pred = results.get_prediction(start=pd.to_datetime('2019-09-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = data.pm10['2019-01':].plot(figsize=[15, 8], label='observed')
pred.predicted_mean.plot(figsize=[15, 8], ax=ax, label='One-step ahead Forecast', alpha=.7)
ax.fill_between(pred_ci.index,
pred_ci.iloc[:, 0],
pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel('pm10 Levels')
plt.legend()
pm10_forecasted = pred.predicted_mean
pm10_truth = data.pm10['2019-09-01':]
# Compute the mean square error
rmse = (((pm10_forecasted - pm10_truth) ** 2).mean()) ** 0.5
print('The Root Mean Squared Error of our prediction is {}'.format(round(rmse, 2)))
forecast = results.forecast(24)
print(forecast)
forecast.plot(figsize=[15, 8], color='green', label='future predictions')
```

```
The Root Mean Squared Error of our prediction is 5.08
2020-01-01 01:00:00    54.143804
2020-01-01 02:00:00    49.976590
2020-01-01 03:00:00    47.240330
2020-01-01 04:00:00    45.095638
2020-01-01 05:00:00    43.519027
2020-01-01 06:00:00    42.427807
2020-01-01 07:00:00    41.895338
2020-01-01 08:00:00    41.844704
2020-01-01 09:00:00    41.122234
2020-01-01 10:00:00    39.872711
2020-01-01 11:00:00    39.684032
2020-01-01 12:00:00    39.801029
2020-01-01 13:00:00    39.491546
2020-01-01 14:00:00    39.223012
2020-01-01 15:00:00    39.049002
2020-01-01 16:00:00    38.912185
2020-01-01 17:00:00    38.811917
2020-01-01 18:00:00    38.743178
```

```
2020-01-01 19:00:00    38.711670
2020-01-01 20:00:00    38.712454
2020-01-01 21:00:00    38.665255
2020-01-01 22:00:00    38.580440
2020-01-01 23:00:00    38.569146
2020-01-02 00:00:00    38.578754
Freq: H, dtype: float64
```

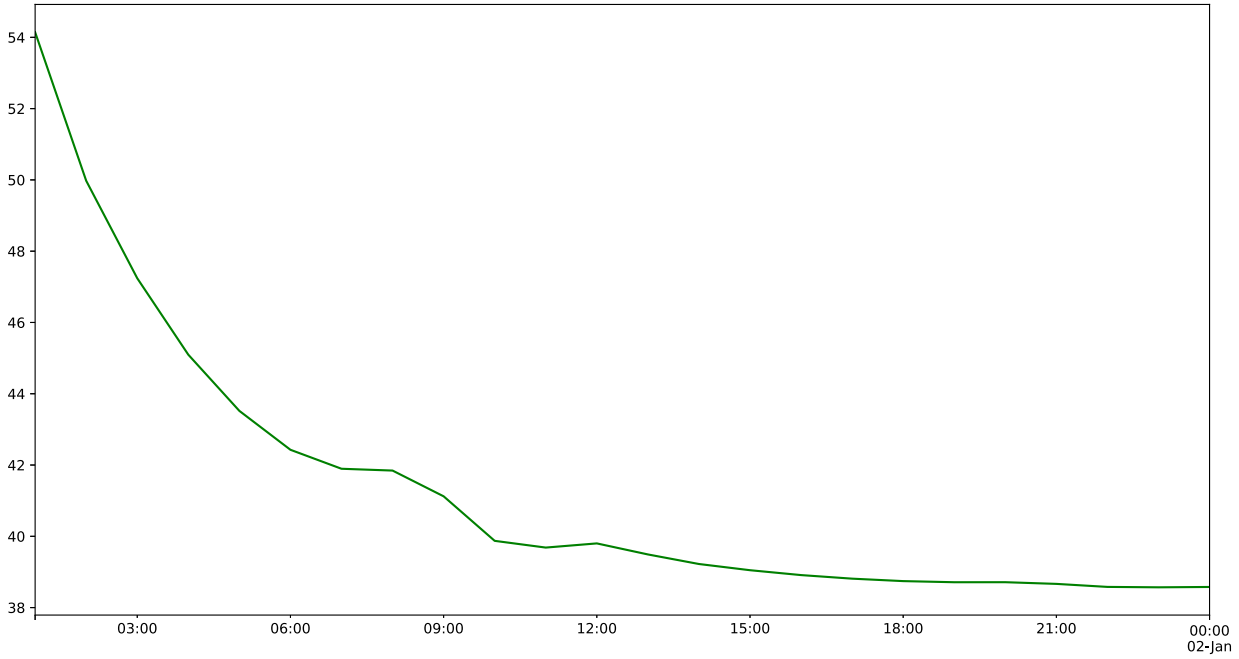
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x2069f5c9940>



In [27]:
```
forecast = results.forecast(24)
print(forecast)
forecast.plot(figsize=[15, 8], color='green', label='future predictions')
```

```
2020-01-01 01:00:00    54.143804
2020-01-01 02:00:00    49.976590
2020-01-01 03:00:00    47.240330
2020-01-01 04:00:00    45.095638
2020-01-01 05:00:00    43.519027
2020-01-01 06:00:00    42.427807
2020-01-01 07:00:00    41.895338
2020-01-01 08:00:00    41.844704
2020-01-01 09:00:00    41.122234
2020-01-01 10:00:00    39.872711
2020-01-01 11:00:00    39.684032
2020-01-01 12:00:00    39.801029
2020-01-01 13:00:00    39.491546
2020-01-01 14:00:00    39.223012
2020-01-01 15:00:00    39.049002
2020-01-01 16:00:00    38.912185
2020-01-01 17:00:00    38.811917
2020-01-01 18:00:00    38.743178
2020-01-01 19:00:00    38.711670
2020-01-01 20:00:00    38.712454
2020-01-01 21:00:00    38.665255
2020-01-01 22:00:00    38.580440
2020-01-01 23:00:00    38.569146
2020-01-02 00:00:00    38.578754
Freq: H, dtype: float64
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x2069cc848e0>


54
52
50
48
46
44
42
40
38
03:00
06:00
09:00
12:00
15:00
18:00
21:00
00:00
02-Jan

In [ ]: