Out[74]:

	proporcion_niños	proporcion_adultos	proporcion_alimento	proporcion_extras	proporcion_total
0	0.206349	0.433862	0.230317	0.060398	0.711224
1	0.192308	0.534965	0.248808	0.060527	0.992992
2	0.197674	0.569767	0.289409	0.054595	0.791973
3	0.276382	0.552764	0.233494	0.062997	1.002248
4	0.149020	0.592157	0.140576	0.053917	0.915151

Método SEM

Variable latente: Satisfacción general por día.

Varoables observables:

- temperatura_promedio
- proporcion_adultos
- proporcion_niños
- proporcion_extras
- proporcion_total
- is_wknd
- is_hs

Si el p-value de alguna variable excede el 5%, esta se remueve del modelo y se repite el proceso hasta no tener variables con p-value mayor al 5%.

Definición del modelo

In [78]:
```
# Definir el modelo SEM usando la notación estándar de SEM
model_desc = """
# Latent Variables
Satisfaccion =~ temperatura_promedio + proporcion_adultos + proporcion_alimento + proporcion_extras + propo
"""
```

In [79]:
```
df_SEM = df_proporciones
df_SEM['temperatura_promedio'] = df['temperatura_promedio']
df_SEM.head()
```

Out[79]:

	proporcion_niños	proporcion_adultos	proporcion_alimento	proporcion_extras	proporcion_total	temperatura_promedi
0	0.206349	0.433862	0.230317	0.060398	0.711224	2
1	0.192308	0.534965	0.248808	0.060527	0.992992	2
2	0.197674	0.569767	0.289409	0.054595	0.791973	2
3	0.276382	0.552764	0.233494	0.062997	1.002248	2
4	0.149020	0.592157	0.140576	0.053917	0.915151	2

Ajuste del modelo

In [81]:
```
numerical_features_SEM = ['temperatura_promedio','proporcion_adultos','proporcion_alimento','proporcion_ex

data_to_model_standarized_SEM = StandardScaler().fit_transform(df_SEM[numerical_features_SEM])

data_to_model_SEM = pd.DataFrame(data_to_model_standarized_SEM,
                                 columns=numerical_features_SEM)

data_to_model_SEM['is_hs'] = df['is_hs']
```

In [82]:
```
mod = Model(model_desc)
res_opt = mod.fit(data_to_model_SEM)
estimates = mod.inspect()

# Imprimir los resultados del ajuste del modelo
estimates
```

Out[82]:

	lval	op	rval	Estimate	Std. Err	z-value	p-value
0	temperatura_promedio	~	Satisfaccion	1.000000	-	-	-
1	proporcion_adultos	~	Satisfaccion	-0.204914	0.07671	-2.671273	0.007556
2	proporcion_alimento	~	Satisfaccion	-0.341163	0.089019	-3.832485	0.000127
3	proporcion_extras	~	Satisfaccion	-0.146549	0.07312	-2.004244	0.045044
4	proporcion_total	~	Satisfaccion	0.301936	0.084942	3.554615	0.000379
5	is_hs	~	Satisfaccion	0.256041	0.054621	4.687599	0.000003
6	Satisfaccion	~~	Satisfaccion	0.738407	0.161676	4.5672	0.000005
7	is_hs	~~	is_hs	0.139787	0.014178	9.859128	0.0
8	proporcion_adultos	~~	proporcion_adultos	0.968885	0.072434	13.37616	0.0
9	proporcion_alimento	~~	proporcion_alimento	0.913817	0.070577	12.947835	0.0
10	proporcion_extras	~~	proporcion_extras	0.983976	0.073118	13.457323	0.0
11	proporcion_total	~~	proporcion_total	0.932448	0.071078	13.118662	0.0
12	temperatura_promedio	~~	temperatura_promedio	0.261667	0.146363	1.78779	0.07381

Resultados del modelo

In [84]:

```
# Cargar factores estimados (cargas factoriales)
confianza_factors = [1.0, -0.204914, -0.341163, -0.146549, 0.301936, 0.256041]  #temperatura, proporcion_a
#proporcion_total, is_hs

# Función para calcular el valor de la variable latente
def calculate_latent_values(df):
    # Calcular Satisfaccion
    df['satisfaccion'] = (df_SEM['temperatura_promedio'] * confianza_factors[0] +
                          df_proporciones['proporcion_adultos'] * confianza_factors[1] +
                          df_proporciones['proporcion_alimento'] * confianza_factors[2] +
                          df_proporciones['proporcion_extras'] * confianza_factors[3] +
                          df_proporciones['proporcion_total'] * confianza_factors[4] +
                          df['is_hs'] * confianza_factors[5])
```

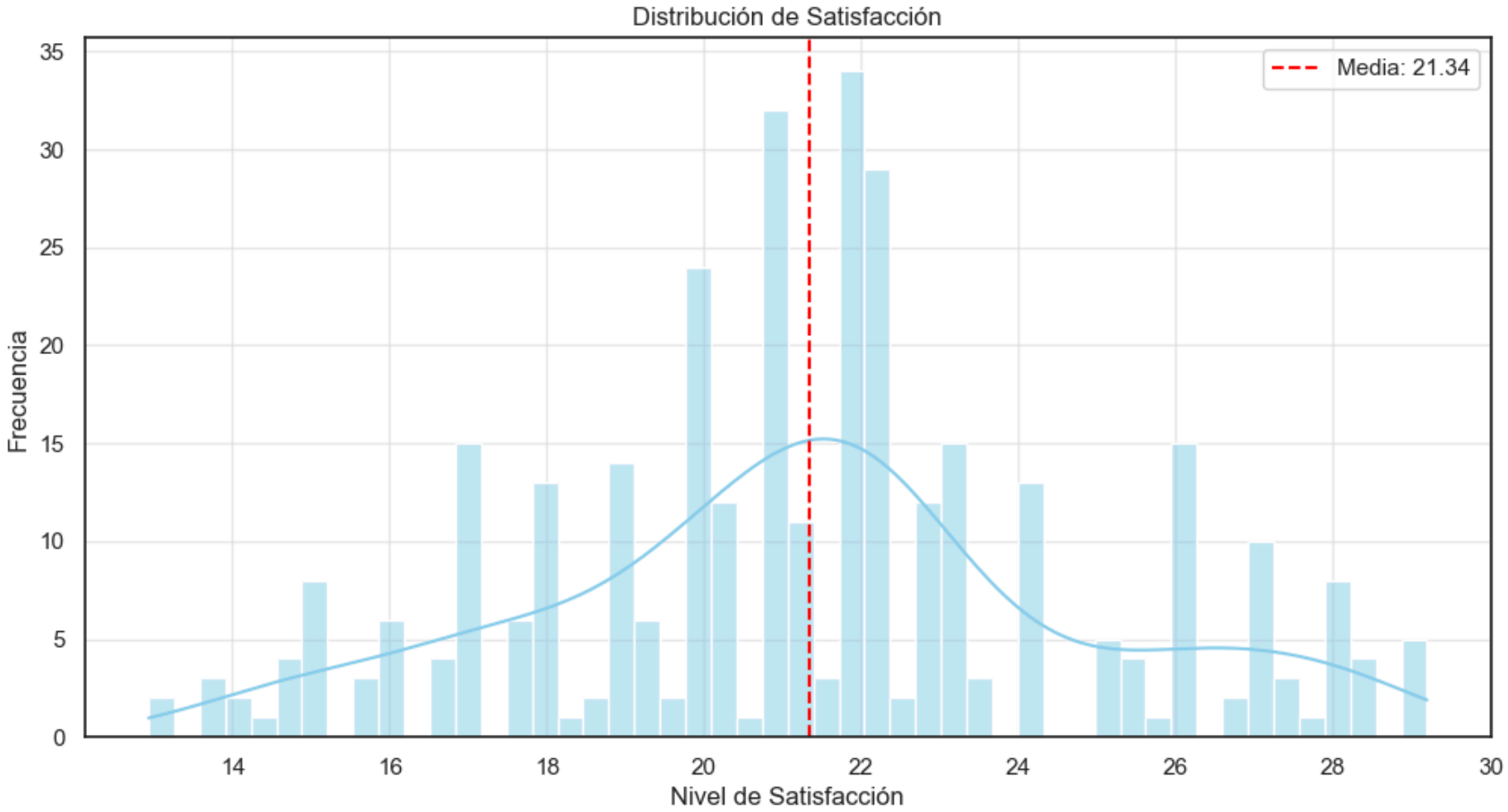
```
    return df[['satisfaccion']]

# Calcular valores latentes en el dataset
latent_values = calculate_latent_values(df)

# Mostrar los primeros valores calculados
latent_values['fecha'] = df['fecha']
latent_values.head()
```

Out[84]:

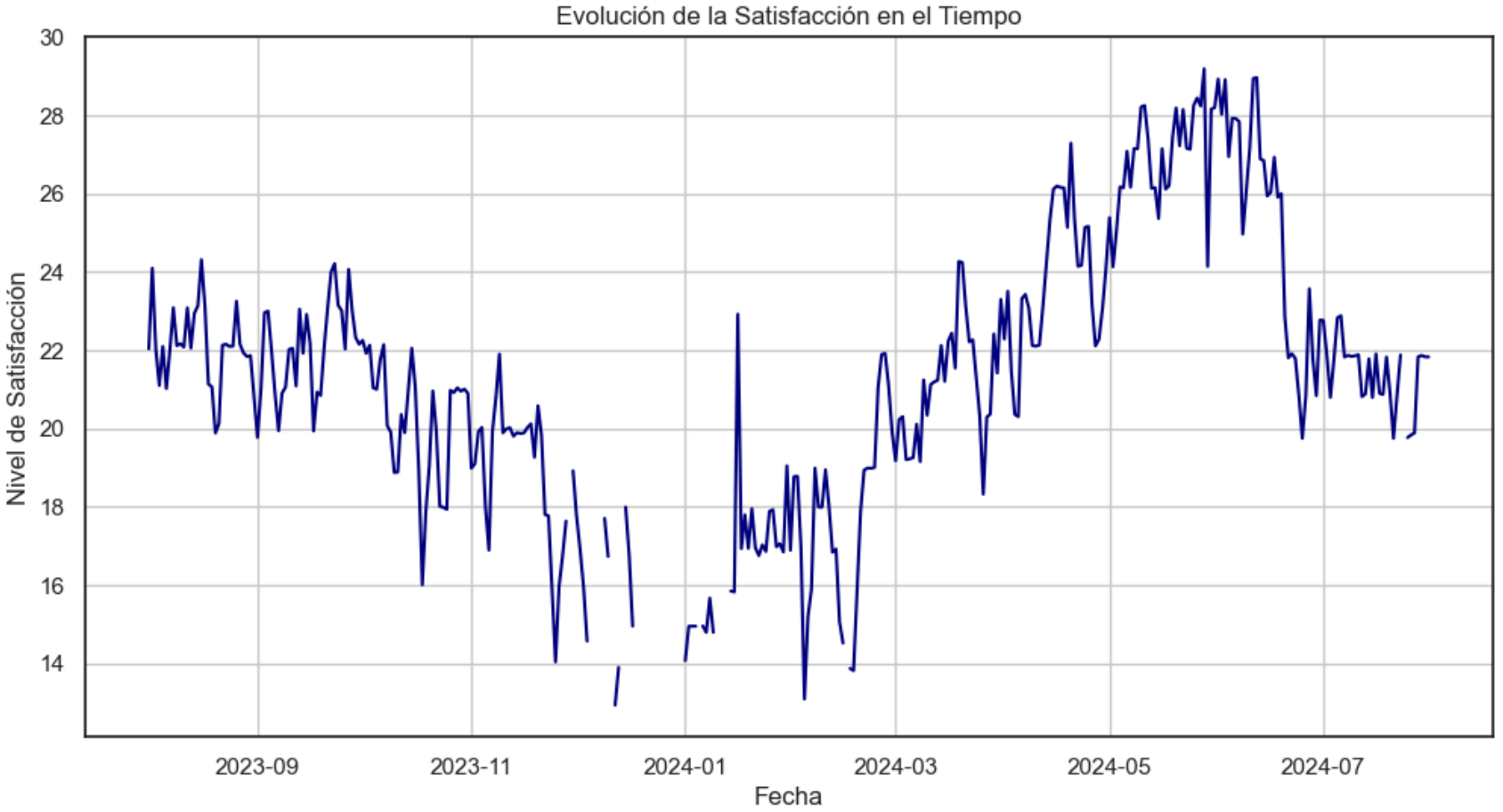
	satisfaccion	fecha
0	22.038413	2023-08-01
1	24.096444	2023-08-02
2	22.015635	2023-08-03
3	21.100454	2023-08-04
4	22.099115	2023-08-05

In [85]:

```
sns.histplot(
    data=latent_values['satisfaccion'],
    bins=50,
    kde=True,  # Añade la curva de densidad
    color='skyblue'
)

sat_mean = latent_values['satisfaccion'].mean()

plt.title('Distribución de Satisfacción')
plt.xlabel('Nivel de Satisfacción')
plt.ylabel('Frecuencia')

plt.axvline(sat_mean, color='red', linestyle='--', label=f'Media: {sat_mean:.2f}')
plt.legend()

plt.grid(True, alpha=0.4)
plt.show()
```



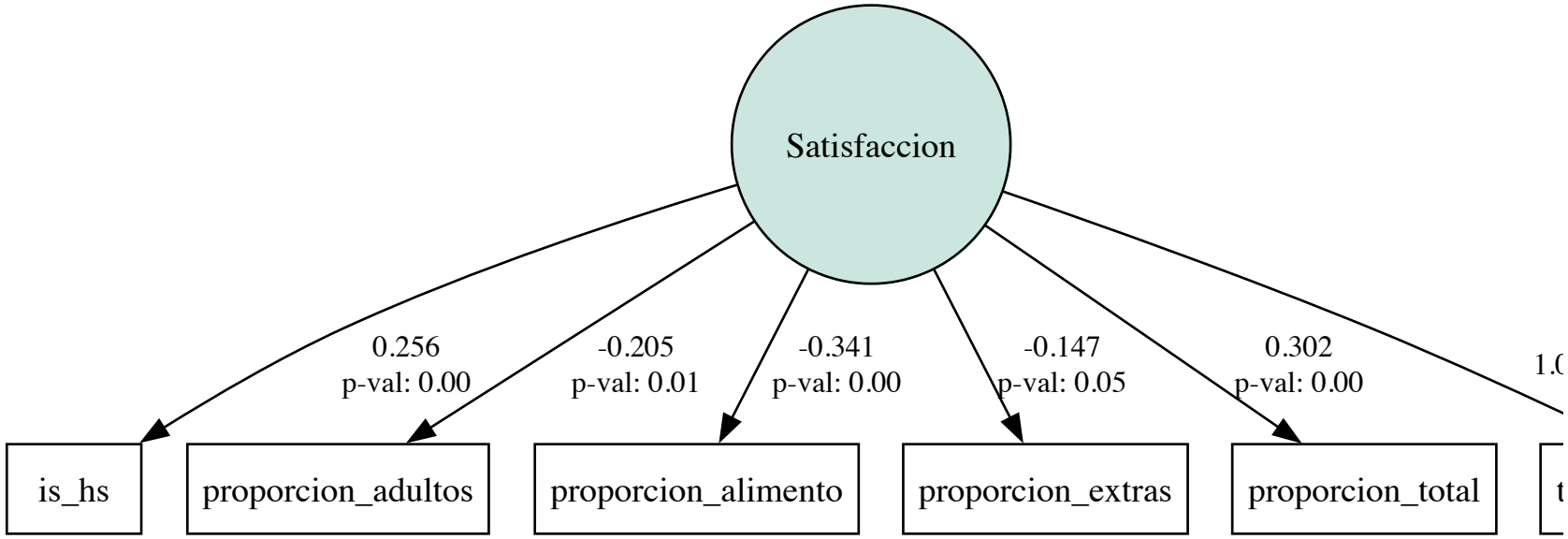
In [86]:
```
plt.plot(latent_values['fecha'],latent_values['satisfaccion'], c = 'navy')

plt.title('Evolución de la Satisfacción en el Tiempo')
plt.xlabel('Fecha')
plt.ylabel('Nivel de Satisfacción')
plt.grid()
```



Visualización del modelo

In [88]:
```
sm = semplot(mod, 'modelo_sem.png', engine = 'dot')
sm
```

Out[88]:



Análisis del modelo

Las variables seleccionadas, tras eliminar aquellas cuyo valor de p-value superaba 0.05, fueron: la temperatura promedio del día, la proporción de adultos respecto al total de visitantes, la proporción de compras adicionales en relación con los ingresos totales, la proporción de visitantes reales en comparación con los esperados y si era temporada alta o no. De estas, se tomó la temperatura promedio como la variable de referencia, asignándole un valor de 1, a partir del cual se compararon las demás.

Las tres variables que más influyeron en la satisfacción general diaria fueron: la temperatura promedio (valor de 1), seguida por la proporción total de asistentes reales contra los esperados(valor de 0.30) y, por último, si era temporada alta (valor de 0.26). También podemos notar que la proporción de adultos, de alimentos y de consumos extra tienen coeficiente negativo, por lo que si esta proporción crece, la satisfacción del día disminuye.

El impacto de estas variables en la satisfacción general se interpreta en función de cuánto afecta la variación de cada una en la ecuación final del modelo. Es decir, un cambio en alguna de estas variables tiene un efecto proporcional en la satisfacción general, según los valores estimados.

Además en la gráfica de la satisfacción en el tiempo se pueden apreciar huecos, estos correponden a los días que el parque recibió 0 visitantes, por lo que el modelo no puede calcular la satisfacción los días que no va nadie, lo cual hace sentido pues

si no van personas al parque no hay una satisfacción que medir.

In [91]:
```
end_time = time.perf_counter()
execution_time = end_time - start_time
execution_time
```

Out[91]:
```
17.426634499999636
```

Created with Jupyter by Luis Márquez, Ana Sofía Hinojosa, and Ivanna Herrera